List decoding of linear block codes

Nielsen, Rasmus Refslund

Publication date:
2001

Document Version
Publisher's PDF, also known as Version of record

List decoding of linear block codes

Ph.D.-Thesis

Rasmus R. Nielsen

September 30, 2001

Abstract

The main theme of this thesis is efficient decoding and list decoding of linear block codes. One of the results is an efficient implementation of recently developed algorithms for list decoding of evaluation codes. As part of this an efficient algorithm for bivariate polynomial interpolation is developed.

Furthermore, existing list decoding algorithms are modified into algorithms for list decoding of several classes of known codes including evaluation codes in the m-metric, concatenated codes, and the newly discovered Xing-Ling codes. In the case of concatenated codes — where calculating the list error correcting capability of the algorithm in general is difficult — the list error correcting capability is calculated explicitly for simple cases and a general method for calculating the list error correcting capability using standard linear programming is found.

In the decoding algorithms erasures are handled transparently by extending distance functions to an alphabet containing a symbol representing an erasure.

Listeafkodning af lineære blokkoder

Resumé

Hovedtemaet for denne afhandling er effektiv afkodning og listeafkodning af lineære blokkoder. Et af resultaterne er en effektiv implementering af nye algoritmer til listeafkodning af evalueringskoder. Herunder er en effektiv algoritme til interpolation af polynomier i to variable udviklet.

Derudover har eksisterende listeafkodningsalgoritmer dannet basis for udviklingen af nye algoritmer til listeafkodning af adskillige kendte klasser af koder omfattende evalueringskoder i m-metrikken, konkatenerede koder og de nye Xing-Ling koder. For konkatenerede koder, hvor det i almindelighed er svært at beregne listeafkodningskapaciteten for algoritmen, er listeafkodningskapaciteten beregnet eksplicit i simple tilfælde og en generel metode baseret på standard lineær programmering til beregning af listeafkodningskapaciteten er fundet.

I afkodningsalgoritmerne bliver slettede symboler håndteret på lige fod med andre symboler ved at udvide afstandsfunktioner til et alfabet indeholdende et symbol som repræsenterer et slettet symbol.

Contents

Preface

The work described in this thesis has been carried out during my enrollment at the Ph.D.-programme in Mathematics at the Technical University of Denmark in the period from September 1st, 1998 to September 30th, 2001 which includes one month of work not related to the Ph.D.-project at the Department of Mathematics. The project has been financed by the Technical University of Denmark and has taken place at the Department of Mathematics with Prof. Tom Høholdt as supervisor.

Partial results of this thesis have been presented at various conferences and seminars and I am grateful for having received financial support for this from the Department of Mathematics, from the Danish Natural Science Research Council, and from the Otto Mønsted Foundation. Furthermore, the Ph.D. project has included visits to the Laboratory of Computer Science at Massachusetts Institute of Technology and Bell Labs Innovations at Lucent Technologies from August to December 2000. I am grateful for having been received with great hospitality in both places, in particular by my hosts, Prof. Madhu Sudan at MIT and Alexander Barg at Bell Labs. I am also grateful for having received financial support for the visits by the Department of Mathematics and by a grant from GN Store Nord Foundation.

Many people have contributed to the results described in this thesis by fruitful discussions, constructive criticism, suggestions, and corrections for which I am very grateful. I am especially indebted to my supervisor, Tom Høholdt, for his extraordinary support at the professional as well as the personal level throughout the period of the project.

Also I am very grateful to my wife Ana for her endless love and support and for always putting a smile in my mind.

This text is typeset with LaTeX 2_ε.

Rasmus R. Nielsen
Lyngby, September 30, 2001

Chapter 1

Algebraic coding theory in brief

This chapter introduces the basic concepts of algebraic coding theory. The description is mostly qualitative with the intention of providing an intuitive model of the applications and limitations of the theory. For details one of the many textbooks on the subject should be consulted, for example [24] or [23].

The main application of algebraic coding theory is in digital communication in a broad sense including for example storage media for digital information. The traditional goal of algebraic coding theory has been to describe and develop the use of algebraic block codes. So far the use of algebraic block codes has been very successful in making digital communication faster and more reliable in a rapidly increasing number of applications.

1.1 Digital communication model

In the general model for digital communication there is a **sender** holding some digital information (supposedly compressed to a level where it contains no redundancy). In this context "digital" means that the information can be represented as a string of symbols from a finite alphabet, A_q, with q elements.

The application of block codes assumes that the information is divided into strings of length k for some integer k. A **(block) code** is then a set of at least q^k strings of length n for some integer n. Elements of this set are called **codewords** and n is the **length** of the code. Each block of the information string is then **encoded** using the block code. This means that each string is mapped to a codeword by a fixed one-to-one mapping. Thus the effect of encoding is essentially that $n-k$ redundant symbols are

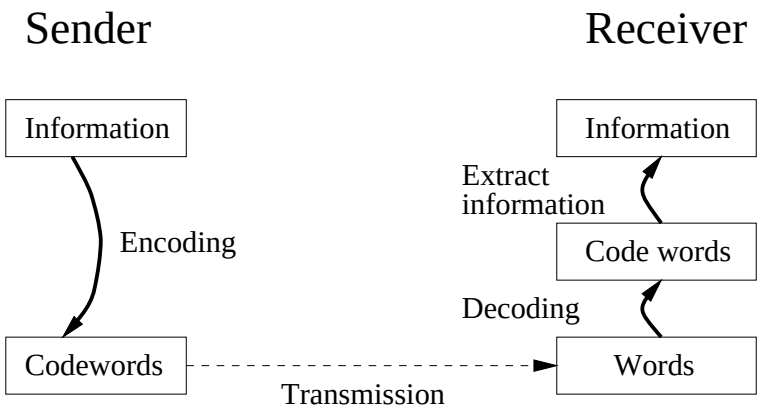

Figure 1.1: *Schematic digital communication model.*

appended to the k information symbols. Therefore, the quantity k/n is called the **information rate** or just the rate of a code.

After the information has been encoded, the codewords are transmitted to the **receiver** over some noisy transmission channel. The noise has the effect that the receiver may not receive exactly the codewords which was sent. Usually it is assumed that the receiver receives **words** of length n over some alphabet, $B \supseteq A_q$. The receiver then uses a **decoder** which ideally maps each received word to the codeword which was most likely the sent one.

This model is shown schematically in Figure 1.1.

1.2 Hamming distance and decoding

The results in this thesis as well as most other results of algebraic coding theory are justified by the following assumption on the digital communication model.

Assumption 1.1
After receiving a word with all its symbols in the input alphabet, A_q, *the receiver may assume that the probability that a certain codeword was sent is strictly decreasing with and depends only on the number of positions where the codeword differs from the received word.* □

This assumption is reflected by the following distance on A_q^n (the set of strings of length n over A_q) which says that the distance between two strings is the number of positions where the strings differ.

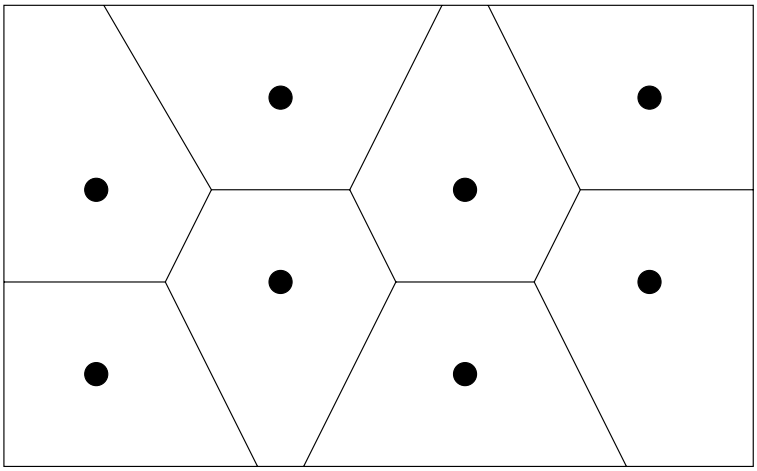

Figure 1.2: *A model of a code with decoding regions. The interior of the box symbolizes all possible words. The points marked by black circles are the codewords and the straight lines form a partition of the box into decoding regions. A maximum likelihood decoder outputs the codeword of a certain decoding region whenever a word is received within that region.*

Definition 1.2 (Hamming distance and weight)
*Let A_q be a set of q symbols and let $n \in \mathbb{N}$ be a positive integer. Then define the **Hamming distance** on A_q^n as the mapping $\mathrm{d} : (A_q^n)^2 \to \mathbb{N}$ given by*

$$\mathrm{d}(u, v) := \{i \mid u_i \neq v_i\}, \quad u, v \in A_q^n.$$

*The **Hamming weight** of some word, $u \in A_q^n$, is given by the distance to the zero word. That is*

$$\mathbf{w}(u) := \mathrm{d}(u, 0).$$

□

With the Hamming distance one may think of a code as a set of points in an n-dimensional box and the (ideal) decoder can be specified as a method which finds a codeword with smallest possible Hamming distance to the received word — by Assumption 1.1 this is a codeword with largest possible probability of being the sent codeword. Turning this around, one may say that for each codeword there is a **decoding region** which consists of the set of words which are closest to that codeword. A decoder which for any received word outputs the codeword of its decoding region is called a **maximum likelihood decoder**. Figure 1.2 shows a model of a code with decoding regions.

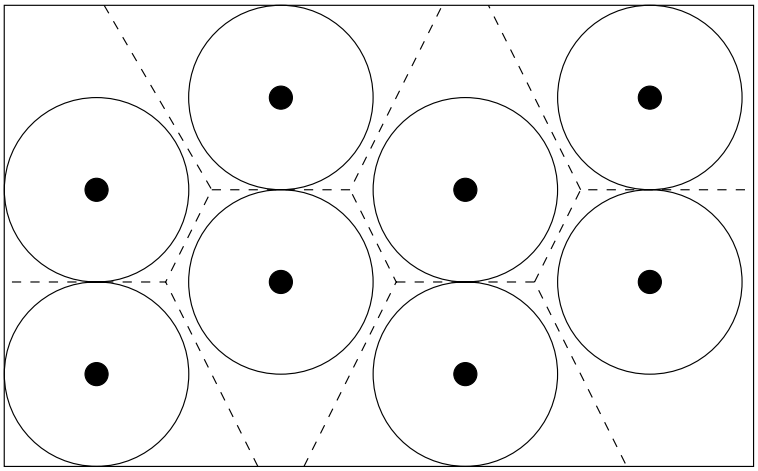

Figure 1.3: *Functionality of a unique decoder. If a received word is within a circle then the codeword at the center of the circle is returned. Otherwise a decoding failure is reported. The maximum likelihood decoding regions are given by the dashed lines.*

A significant limitation of algebraic coding theory is that maximum likelihood decoders generally require either a time of computation or an amount of memory which is not available in practice. Instead, most decoders have the following functionality for some $t > 0$ called the **error correcting capability** of the decoding method: If a codeword exists (strictly) within distance t from the received word then such codeword is returned. Otherwise, a **decoding failure** is reported.

Given a code, the **minimum (Hamming) distance** is defined as the smallest distance between two different codewords. Let this be denoted by d. If $t \leq d/2$ then the output of a decoder with error correcting capability t is uniquely determined. Such a decoder will be called a unique decoder. In practice, almost all efficient decoders are unique decoders with functionality as shown in Figure 1.3. Notice that many received words result in a decoding failure and even though these words are far from codewords and thus have small probability of occurring, the use of a unique decoder generally gives a significant reduction (compared to a maximum likelihood decoder) in the amount of information that can be transmitted using a given channel.

It should be noted that the minimum distance is generally the most important measurement of the quality of a code and thus a major topic in algebraic coding theory is to construct codes with a minimum distance which is as large as possible for a given alphabet, length, and number of

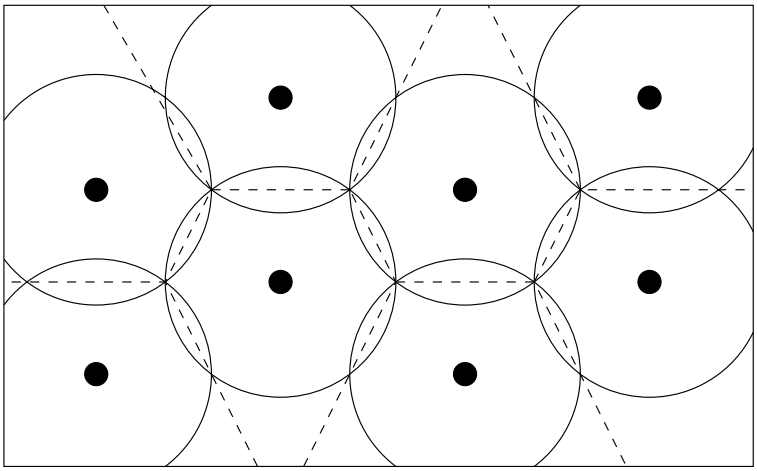

Figure 1.4: *List decoding. When a word is received the list decoder returns the set of codewords within a specified distance (the radius of the circles) from the received word. The maximum likelihood decoding regions are given by the dashed lines.*

codewords.

A useful generalization of the decoding problem is to determine the set of all codewords within some distance, τ, from the received word. Such a decoder is called a **list decoder** and provided that the number of codewords in the output is small, a list decoder can easily be turned into a traditional decoder with error correcting capability τ by picking any codeword in the output which is closest to the received word. The functionality of a list decoder is shown in figure 1.4.

Much of the work described in this thesis has been dedicated to develop new efficient unique decoders and list decoders.

1.3 The output alphabet

Whenever the output alphabet, B, is different from the input alphabet, A_q, then some extension, $\mathrm{d}^* : (B^n)^2 \to \mathbb{R}$, of the Hamming distance is needed in order to specify the desired functionality of the decoder. Since d^* is an extension of the Hamming distance the minimum d^*-distance of a code equals its minimum Hamming distance, d. Thus decoding is unique for received words within d^*-distance $d/2$ from a codeword.

Notice that any symbol in A_q is characterized by having distance 0 to itself and distance 1 to any other symbol in A_q (the distance here is the Hamming distance for $n = 1$). Similarly, any additional symbol in B can

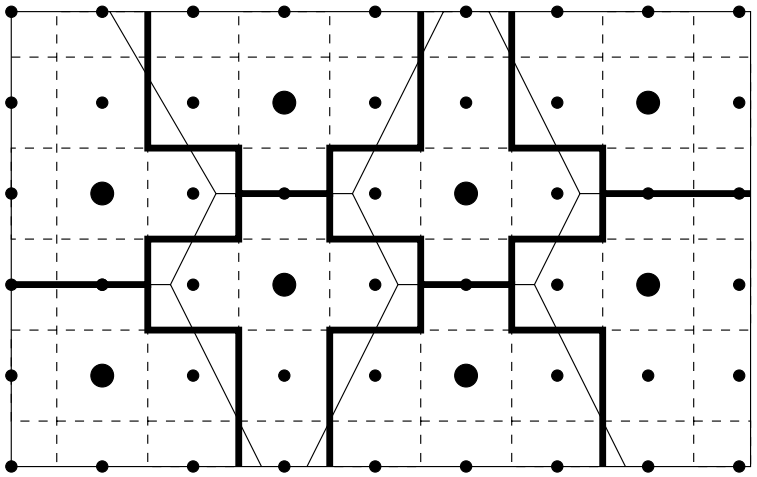

Figure 1.5: *Decoding with a limited output alphabet. The dots symbolize words with elements in the output alphabet. Decoding is done by first identifying the dot closest to the received word (the decoding regions are shown by the dashes lines) and then decode to the closest codeword (black circle). The thin lines mark the decoding regions of a maximum likelihood decoder and the thick lines show the overall decoding regions of the two step decoder.*

be characterized by the set of its extended distances to the symbols in A_q. It is common to have a symbol, ?, in B which has extended distance $1/2$ to any symbol in A_q. This symbol is called an erasure and represents a received value that could not be matched to any other symbol in B.

Efficient algebraic decoders usually cannot handle additional symbols in B besides erasures. Therefore, upon receiving a word the received symbol on each of the n positions must be matched with a symbol of A_q or an erasure must be placed. After this the usual decoding is done. This is shown in figure 1.5. Notice that matching the received word to a word with elements in the output alphabet may imply a move into a different decoding region. The effect is that the decoding regions of the overall decoding differs from the maximum likelihood decoding regions and thus the output is more likely to be erroneous than with maximum likelihood decoding. Clearly, the difference increases with a smaller output alphabet.

Actually, Figure 1.5 illustrates a common situation for composed decoders. Such decoders occur often since many codes are composed from two component codes with the purpose of being able to construct a decoder in terms of decoders for the component codes. Often much care is needed in order to obtain the desired error correcting capability because

the result of one component decoder easily leads the other decoder in a wrong direction.

Having a small output alphabet (either $B = A_q$ or $B = A_q \cup \{?\}$) is a significant and almost general limitation of algebraic coding theory, however it is interesting that recently developed list decoding techniques — which also are the basis of most of the results of this thesis — makes it possible to treat more general output alphabets.

1.4 Linear codes

In order to be able to analyze codes and — equally important — in order to be able to encode information and decode received words in an efficient manner, codes are usually constructed having some algebraic structure. Most algebraic codes have an alphabet which is a **finite field** of q elements, denoted by $\mathbb{F}_q$, which implies that $\mathbb{F}_q^n$ is a vector space over $\mathbb{F}_q$. Furthermore, most algebraic codes are **linear** which means that they form a k-dimensional subspace of $\mathbb{F}_q^n$.

For linear codes encoding can be done efficiently by multiplying the information vector (any element of $\mathbb{F}_q^k$) by a $k \times n$ **generator matrix** whose rows form a vector space basis of the code. The set of (column) vectors which when multiplied on the generator matrix gives an all-zero vector forms a $n-k$ dimensional subspace of $\mathbb{F}_q^n$. A **parity check** matrix is a $(n-k) \times n$ matrix whose rows form a basis of this vector space.

The vector obtained by multiplying a word (an element of $\mathbb{F}_q^n$) on the parity check matrix is called the **syndrome** of the word. The set of codewords is then exactly the words whose syndrome is the all-zero vector. This gives an efficient method for detecting errors. Furthermore, each syndrome can be associated with a minimal error pattern which is a word of minimal weight which has the given syndrome. This gives a time efficient decoding method (calculate the syndrome of a received word, find the corresponding minimal error pattern in a precalculated table, and the error pattern from the received word) which is very good for codes with high information rate (few different syndromes). However, in general the method is not space efficient so it is only useful for small codes.

The following gives an example of a linear code and syndrome decoding:

Example 1.3

A binary code is a code over the alphabet $\{0,1\}$ *which forms the finite field* $\mathbb{F}_2$*. The following is a generator matrix,* G*, and a parity check matrix,* H*, of a binary code of length* 7*, dimension* 4*, and minimum distance* 3*:*

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Suppose that this code is used for communication and that the word $r = (1,0,1,1,0,0,0)$ *is received. The syndrome is* $Hr^T = (0,1,0)^T$ *so the received word is not a codeword. However, the syndrome equals the sixth column of the parity check matrix. Therefore,*

$$c = r + (0,0,0,0,0,1,0) = (1,0,1,1,0,1,0) = (1,0,1,1) \cdot G$$

is a codeword of distance 1 to the received word. Thus, c *should be the output of the decoder.* □

1.5 Evaluation codes

Evaluation codes are usually constructed using a k-dimensional $\mathbb{F}_q$ vector space of (polynomial) functions and a fixed tuple of mutually distinct elements from the common domain of the functions. There is a one-to-one correspondence — called an **evaluation map** — between functions and codewords where a codeword is obtained by evaluating the given function in each element of the tuple. If the evaluation map is linear then the code is linear.

One useful property of linear evaluation codes is that any upper bound on the number of zeroes that a function in the vector space may have among the elements of the tuple implies a lower bound on the minimum Hamming distance of the code. Furthermore, the algebraic structure on polynomial functions given by the multiplication of polynomials is often useful in finding efficient decoding methods. The decoding is usually based on finding the polynomial corresponding to the correct codeword by interpolation through the received points with some mechanism to ignore erroneous values — an example of this is given in the next section.

Many of the most studied and applied algebraic block codes are evaluation codes and most of the results described in this thesis are on evaluation codes. Examples of evaluation codes are Reed-Solomon codes, Reed-Muller codes, and algebraic geometry codes.

1.6 Decoding example (ad hoc)

This section gives an example of an ad hoc method for decoding a given evaluation code (a Reed-Solomon code).

Let the alphabet be $\mathbb{F}_7 = \mathbb{Z}_7$ — the integers $\{0, 1, 2, 3, 4, 5, 6\}$ with addition, subtraction, and multiplication modulo 7. Consider the set of all vectors obtained from evaluating the elements of $\mathbb{Z}_7$ in all polynomials of degree at most 2 (including the zero polynomial):

$$\{(f(0), f(1), \ldots, f(6)) \mid f \in \mathbb{Z}_7[x] \wedge \deg(f) \leq 2\}. \tag{1.1}$$

This is a Reed-Solomon code of length 7, dimension 3, and minimum distance 5. In this case there are $7^4 = 2401$ different syndromes so decoding with syndromes would be cumbersome.

Instead, suppose that a word, $r = (r_0, \ldots, r_6)$ is received and that at most 2 errors occurred. Notice that if we pick three mutually distinct values, $t, u, v \in \mathbb{Z}_7$, then there is a unique polynomial, $f := f_{t,u,v}$ of degree at most 2 satisfying $f(t) = r_t$, $f(u) = r_u$, and $f(v) = r_v$. This can be calculated as follows.

Let f be written as

$$f = f_2x^2 + f_1x + f_0 = f_2'(x-t)(x-u) + f_1'(x-t) + f_0'$$

for suitable values $f_0, f_1, f_2, f_0', f_1', f_2' \in \mathbb{Z}_7$. Then

$$\begin{aligned} f(t) &= f_0' \\ f(u) &= f_1'(u-t) + f_0' \\ f(v) &= f_2'(v-t)(v-u) + f_1'(v-t) + f_0' \end{aligned}$$

and finding f_0', f_1', and f_2' is straightforward:

$$\begin{aligned} f_0' &= f(t) \\ f_1' &= \frac{f(u) - f_0'}{u-t} \\ f_2' &= \frac{f(v) - f_1'(v-t) - f_0'}{(v-t)(v-u)} = \frac{f(v) - f(t)}{(v-t)(v-u)} - \frac{f(u) - f(t)}{(u-t)(v-u)}. \end{aligned}$$

Now f_0, f_1, and f_2 can be calculated:

$$\begin{aligned} f_0 &= f_0' - tf_1' + tuf_2' \\ f_1 &= f_1' - (t+u)f_2' \\ f_2 &= f_2' \end{aligned}$$

If r_t, r_u, and r_v are all correct then $f_{t,u,v}$ is the polynomial corresponding to the correct codeword. Suppose that $f_{t,u,v}$ is calculated for the 5 assignments of t, u, and v given by the table below (t, u, and v are assigned the values marked by an X in each attempt).

Attempt	0	1	2	3	4	5	6
1	X	X	X				
2				X	X	X	
3	X			X			X
4		X			X		X
5			X			X	X

It can be checked that for any pattern of 0, 1, or 2 errors which may occur the interpolation in at least one of the 5 attempts will be with 3 correct values (for any pair of columns there is always at least one row which does not have an X in either of the columns). Thus the correct polynomial is obtained in one of the attempts. The correct polynomial can be identified since it corresponds to a codeword within distance 2 of the received word. If none of the 5 polynomials satisfy this then more than 2 errors occurred and a decoding failure is declared.

The following gives an example of using this decoding method.

Example 1.4
Suppose that the Reed-Solomon code of (1.1) is used for communication and that

$$r = (5, 4, 1, 5, 6, 2, 6)$$

is received.

First, we find $f \in \mathbb{Z}_7[x]$ *such that* $\deg(f) \leq 2$ *and* $f(0) = 5$, $f(1) = 4$, *and* $f(2) = 1$. *This gives* $f = 6x^2 + 5$ *corresponding to the codeword* $(5, 4, 1, 3, 3, 1, 4)$ *which has distance* 4 *to the received word.*

Therefore, we continue and find $g \in \mathbb{Z}_7[x]$ *such that* $\deg(g) \leq 2$ *and* $g(3) = 5$, $g(4) = 6$, *and* $g(5) = 2$. *This gives* $g = x^2 + x$ *which corresponds to the codeword* $(0, 2, 6, 5, 6, 2, 0)$ *having distance* 4 *to the received word.*

So we find $h \in \mathbb{Z}_7[x]$ *such that* $\deg(h) \leq 2$ *and* $h(0) = 5$, $h(3) = 5$, *and* $h(6) = 6$. *This gives* $h = 2x^2 + x + 5$ *corresponding to the codeword* $c = (5, 1, 1, 5, 6, 4, 6)$. *This has distance* 2 *to the received word, so we stop and give* c *as the decoding result.* □

The decoding method described in this section can be implemented efficiently, but it is not very flexible and it is not clear how erasures should be handled. So the decoding methods described in this thesis will be different, however the example shows how the decoding problem can be turned into a problem of polynomial interpolation. Furthermore, several of the decoders encountered in this thesis will be based on the same strategy of performing a number of decoding attempts until an acceptable output is obtained — alternatively, a limited number of attempts are performed and the result closest to the received word is given as output.

Chapter 2

Overview

Seven papers have been written as part of the Ph.D.-project. Each of these are included as a chapter in this thesis with minimal changes. However, the layout has been changed to give a homogeneous layout and references to other papers are gathered in the common bibliography of the thesis. References between the papers has been changed to local references within the thesis.

This chapter gives a short description and a summary of results and related results for each paper.

2.1 Decoding Reed-Solomon codes beyond half the minimum distance

This paper is joint work with T. Høholdt. The paper was published in [28] which is the proceedings of a conference held in Guanajuato, Mexico, in April 1998. Some of the results of the paper were presented at the conference. Furthermore, some of the results appeared in my Master's Thesis [26].

This paper describes explicitly an efficient implementation of Sudan and Guruswami's improved list decoding algorithm for Reed-Solomon codes [13]. An implementation based on the description of the paper was done in C++ and the output of this implementation was used to generate the example in the paper.

The paper also describes how an upper bound on the probability of getting more than one candidate in the output when doing list decoding up to a given distance. The calculation does not depend on the specific list decoding algorithm, nor on the code in use, however, the code is assumed to

be linear and have a known weight distribution. Even though the method of calculation was found independently and had not been applied in connection with Sudan's list decoding algorithm, the method turned out to be a well-known method for calculating the probability of erroneous decoding.

The implementation of the list decoding algorithm, however, was novel. In the first step — the interpolation step — a generalization of a well-known algorithm for polynomial interpolation (an application of the Fundamental Iterative Algorithm [7]) was used in order to find a polynomial with zeroes of a given multiplicity in a given list of points. A slightly optimized version of this algorithm (described in Chapter 5) is still the fastest method available for implementing this step of the list decoding algorithm.

The second step — the factorization step — is a bivariate factorization where only factors in the form $y - f(x)$ with $\deg(f) < k$ are needed. This was implemented by embedding polynomials in $\mathbb{F}_q[x, y]$ in $\mathbb{F}_{q^k}[y]$ and thereby turn the bivariate factorization problem into a univariate problem over a large field. The univariate factorization over a large finite field can be solved using a probabilistic algorithm due to Berlekamp [12, Chapter 8]. This method is faster than doing a bivariate factorization, however, simultaneously, Roth and Ruckenstein [33, Figure 2] proposed to use so-called Newton interpolation in this step of the algorithm, which is faster and gives a deterministic algorithm. Another description is given by Augot and Pecquet [1].

It has also been proposed to use Newton polygons in the Newton interpolation (to avoid iterations at coefficients which are 0). However, in the context of list decoding the polynomials $f(x)$ in general are dense so the use of Newton polygons does not give a faster method. On the contrary, the method is likely to be slower because of the computations spent on the Newton polygon.

Gao and Shokrollahi [11] obtain a faster method by doing all calculations modulo x^k.

2.2 Decoding Hermitian codes with Sudan's algorithm

This paper is joint work with T. Høholdt. A shortened version was published in [16], which is the proceedings of the 13th International Symposium, AAECC-13, held in Honolulu, Hawaii, Nov. 1999. The paper was

prepared prior to the conference and the results were presented at the conference. Results from this paper were also presented at the Winter School on Coding and Information Theory in Ebeltoft, Denmark, Dec. 1998. The paper included in this thesis is the full version containing proofs of the theorems and an example.

The paper explicitly describes an implementation of the list decoding algorithm of Guruswami and Sudan [13] for Hermitian codes. However, the principles of the implementation can be applied to any one-point algebraic geometry codes. The paper contains non-trivial generalizations of the interpolation step and the factorization step implemented in the paper "Decoding Reed-Solomon Codes beyond Half the Minimum Distance". Furthermore, an explicit method for finding a so-called increasing zero basis in the Hermitian function field is developed.

A generalization of the Newton interpolation method was later done by Wu and Siegel [41] which gives a faster method for the factorization step.

An explicit expression for increasing zero basis in the Hermitian function field is given in the paper "Fast bivariate interpolation" (Chapter 5) which means that the method for calculating these increasing zero basis is now obsolete. However, the principles of the method may be applied to other function fields where no explicit expression is known.

2.3 Fast bivariate interpolation

This paper was written mainly for this thesis. It was completed in August 2001 and at the time of writing it has not been published nor submitted for publication.

The main result of the paper is to make a unified version of the algorithm used to implement the interpolation step of various decoding algorithms building on the principles of the list decoding algorithm of Guruswami and Sudan [13]. In the paper the algorithm is stated for polynomials with coefficients in certain $\mathcal{L}$-spaces of a algebraic function field. The proof of the correctness of the algorithm fixes the proofs of correctness of the special cases of the interpolation algorithm of the papers "Decoding Reed-Solomon Codes beyond Half the Minimum Distance" and "Decoding Hermitian Codes with Sudan's Algorithm" both of which lack some details.

Furthermore, the paper contains descriptions of the algorithm in the

rational function field and in the Hermitian function field where the special structure of the function field in each case is used to write the algorithm in a more explicit way than possible in the general case. This includes an explicit expression for so-called increasing zero bases in the Hermitian function field.

The algorithm described in the paper is general enough to be applied to all presently known variants of Guruswami-Sudan's list decoding algorithm. This includes the algorithms described in the other papers of this thesis.

2.4 A class of Sudan-decodable codes

This paper was published in [27]. Preliminary results from the paper were presented at the International Meeting on Coding Theory and Cryptography, Medina, Spain in September 1999 and at the seminar Arithmetique, geometrie et theorie des codes, Luminy, France, October 1999. Furthermore, results from the final version of the paper were presented at the 2000 IEEE International Symposium on Information Theory, Sorrento, Italy in June 2000.

The paper describes a class of codes which was found by analyzing which type of codes could be decoded using Sudan and Guruswami's list decoding algorithm. The result is a construction similar to that of Reed-Solomon codes and Algebraic-Geometry codes when constructed as evaluation codes, however the evaluation in each point is done using a generalized evaluation where r field elements are obtained at each evaluation point for a given positive integer r. The distance properties of these codes are best described using a distance different from the Hamming distance, namely the r-distance (the minimum Hamming distance is generally bad).

When presenting these results at the seminar in Luminy it turned out that Rosenbloom and Tsfasman at the very same seminar a couple of years earlier had presented exactly the same results under the names m-metric, Reed-Solomon m-codes, and Algebraic-Geometry m-codes. These are described in [32].

However, Rosenbloom and Tsfasman do not consider decoding so the decoding algorithms for unique decoding and list decoding which are described in the paper are new. Furthermore, it is shown that Reed-Solomon m-codes can be encoded systematically in any set of positions satisfying

certain requirements — this is the m-metric equivalent of the theorem saying that a Reed-Solomon code of dimension k can be systematically encoded in any k positions.

The paper also gives examples of decoding and performance comparisons between Reed-Solomon m-codes and usual Reed-Solomon codes and between Hermitian m-codes and usual Hermitian codes. From the comparisons it seems probable that codes for the m-metric only perform better than their Hamming-metric equivalents in situations where it can be said in advance that symbols on some given positions have larger error probability than others.

The number of applications where codes in the m-metric might be used seems rather limited at this moment. However, in particular Reed-Solomon m-codes have a very regular structure and may occur implicitly or explicitly in results on codes in the Hamming metric. Examples of this are found in Chapter 8 where a generalization of Reed-Solomon m-codes is used in the decoding of codes in the Hamming metric and in Chapter 9 where a new class of codes for the Hamming metric is constructed.

2.5 Decoding concatenated codes with Sudan's algorithm

The first version of this paper was submitted for publication in IEEE Transactions on Information Theory in June 2000. The review reports were in principle positive, but had several suggestions. The revised — and much expanded — version was submitted in July 2001.

The paper defines an extension of the Hamming distance in order to describe decoding with erasures in a transparent way. Furthermore, the so-called badness of an error pattern is defined. The badness is at most the weight of the pattern and it is shown that Forney's Generalized Minimum Distance (GMD) decoding [10] can be used to decode all error patterns of badness less than half the designed minimum distance of a linear concatenated code. — It is well-known that GMD decoding can be applied to decode all error patterns of weight less than half the designed minimum distance, however, the result of the paper includes a larger class of error patterns and the proof introduces the method of analysis used in the rest of the paper.

In the description of Guruswami and Sudan's list decoding algorithm it is shown that the maximal error correcting capability is obtained by choosing the zero multiplicity (a parameter of the algorithm) to be of the same order of magnitude as n^2 (to be $O(n^2)$) where n is the length of the code. This is well-known. However, it is also shown that choosing a zero multiplicity of $O(n)$ is sufficient to obtain an error correcting capability which is at most 1 less than the maximal error correcting capability — regardless of the length. Choosing a zero multiplicity of $O(n^\gamma)$ for an arbitrary small real number $\gamma > 0$ is sufficient to obtain the full asymptotic fractional error correcting capability (which means that in that case the difference between the maximal and the obtained error correcting capability divided by n vanishes as n tends to infinity).

The main result of the paper is to analyze the error correcting capability of a list decoder for linear concatenated codes composed by some inner decoder and the list decoding algorithm of Guruswami and Sudan [13] as outer decoder. The Guruswami-Sudan algorithm is invoked with zero multiplicities varying at each position in a specified way according to the output of the inner decoder. The analysis is done using linear programming techniques.

First, it is assumed that the inner decoder decodes error patterns if and only if the weight is less than half the minimum distance of the inner code. Under this assumption some tight upper bounds are shown on the error correcting capability of *any* list decoder with reasonably small list size.

When the minimum distance of the inner code is even (or when erasure decoding is needed) this upper bound equals half the designed minimum distance and it is shown that this error correcting capability is obtained by the overall decoder when using Guruswami and Sudan's list decoder as the outer decoder.

When the minimum distance of the inner code is odd and erasure decoding is not needed a method for calculating the error correcting capability (or a good lower bound) of the list decoder is presented. Furthermore, an asymptotic bound on the fractional (with respect to the length of the outer code) list error correcting capability of the decoding method is given for *any* choice of zero multiplicities. Furthermore, the asymptotic fractional list error correcting capability of the decoding method is found for the proposed choice of zero multiplicities and it is seen that this error correcting

capability approaches one of the shown upper bounds for the information rate of the outer code tending to 0 as well as for the information rate of the outer code tending to 1. In all cases the asymptotic fractional list error correcting capability is strictly larger than half the fractional designed minimum distance.

Next, it is assumed that the inner decoder is a list decoder giving the set of codewords within a given distance from the received word. In this case the error correcting capability of the overall list decoder depends on the coset weight distribution of the inner code. Provided that this is known, a method for calculating a good lower bound on the error correcting capability of the list decoder is developed and an example is given where the list error correcting capability is calculated for some given concatenated codes.

Independently, Guruswami and Sudan [14] described the use of the same decoding method, but only in the case where the inner code is a Hadamard code and only stating asymptotic results on the error correcting capability. It should be noted that in [14] the results are also valid when the inner code is a first order generalized Reed-Muller code (this reduces the code length with a factor q for any given dimension).

2.6 Decoding Xing-Ling codes

The first version of this paper was submitted in February 2001 for publication in IEEE Transactions on Information Theory. The reviews were positive regarding the results of the paper, however some suggestions were made on the presentation. The second version was significantly expanded and submitted in September 2001.

Xing-Ling codes [42] form a recently invented class of codes which resulted in several improvements to Brouwer's table [4] of best known linear codes. This paper describes the construction (basically punctured subfield subcodes of Reed-Solomon codes constructed as evaluation codes) and prove the same lower bound on the minimum distance as the original paper. However, the derivation is done in a unified way combining four theorems of the original paper into one. Furthermore, it is suggested to lengthen the codes by evaluating in the point at infinity on the projective line. This is a well-known method for lengthening Reed-Solomon codes and when applied to Xing-Ling codes a number of improvements to Brouwer's

table are obtained.

The main result of the paper is a decoding method for Xing-Ling codes with error correcting capability equal to half the designed minimum distance. The idea of the method is to see a Xing-Ling code as a punctured subfield subcode of two different codes (supercodes) and then decode a received word using both codes. If the weight of the error and erasure pattern is less than half the designed minimum distance then it is proved that at least one of the attempts recovers the correct codeword.

One of the supercodes is described as a generalized Reed-Solomon m-code and a generalized m-metric is defined accordingly. Furthermore, an extension to this metric is defined in order to describe decoding with erasures under the (generalized) m-metric. Also a modification of the algorithm of Chapter 6 is given for unique decoding and list decoding of generalized Reed-Solomon m-codes with errors and erasures.

Furthermore, the asymptotic fractional designed minimum distance of Xing-Ling codes is found and the asymptotic fractional error correcting capability of a proposed list decoder is calculated. A list decoder based on the unique decoder is proposed and the error correcting capability is found in terms of error correcting capabilities of the component decoders.

2.7 Unequal error protection and the m-metric

This paper was completed in August 2001. It is not planned to publish the paper outside this thesis since the results are not likely to be useful. However, the paper has been included in this thesis since the investigation can be seen as an application of methodology and results developed in previous papers (Chapter 7 and Chapter 8). Thus the paper supports the generality of the approach and definitions of these papers.

The paper describes a new construction of codes composed of a code with unequal error protection and a code designed for the m-metric. A lower bound on the minimum Hamming distance in terms of the distance properties of the component codes is proved. However, a non-existence theorem on linear codes with unequal error protection makes it almost certain that the composed codes from this construction will always be inferior to codes constructed from other constructions, in particular product codes.

A decoding method for error and erasure decoding of the composed

construction is found in terms of decoders for the component codes. It is worth noticing that the composed decoder achieves an error correcting capability of half the designed minimum distance using only one decoding attempt — usually composed decoders need several attempts and/or advanced bookkeeping in order to achieve an error correcting capability of half the designed minimum distance.

Chapter 3

Decoding Reed-Solomon codes beyond half the minimum distance

R. Refslund Nielsen and T. Høholdt[1]

Abstract

We describe an efficient implementation of M. Sudan's algorithm for decoding Reed-Solomon codes beyond half the minimum distance. Furthermore, we calculate an upper bound of the probability of getting more than one codeword as output.

3.1 Introduction

In a recent paper M. Sudan [38] presented an algorithm for correcting more than $\frac{d_{\min}-1}{2}$ errors in a Reed-Solomon code with low rate and in [13] he extended his algorithm to higher rates. The algorithm produces a list of codewords closest to the received word. In this paper we present an efficient implementation of Sudan's extended algorithm by speeding up the two crucial steps, namely interpolation and factorization. Based on the weight distribution of MDS codes we also calculate an upper bound on the probability that the list contains more than one codeword. The paper is organized as follows: Section 2 contains some basic definitions and in Section 3 we present Sudan's extended algorithm and prove that it works. Section 4 calculates the asymptotic error-correcting capability of the algorithm and Section 5 gives an upper bound on the number of candidates. Section 6 contains the efficient method to get the interpolation

[1]Technical University of Denmark, Department of Mathematics, Bldg 303, DK-2800 Lyngby, Denmark

polynomial, this is partly based on R. Kötter [20], and Section 7 is devoted to the factorization. Finally we give some examples in Section 8 and Section 9 contains the conclusion and some remarks.

3.2 Basic definitions

Let $\mathbb{F}_q$ denote a finite field with q elements. If $a, b \in \mathbb{F}_q^n$ then $\mathrm{d}(a, b)$ will denote the Hamming-distance between the words a and b. The Hamming-weight of a word, a, will be denoted by $\mathbf{w}(a)$.

Definition 3.1 (Reed-Solomon codes)
Let $P = \{P_1, \ldots, P_n\} \subseteq \mathbb{F}_q$ with $|P| = n$. For $k \leq n$ the set

$$\mathrm{RS}(P, k) = \{f(P) \mid f \in \mathbb{F}_q[x] \wedge \deg(f) < k\}$$

*where $f(P) = (f(P_1), \ldots, f(P_n))$ is called a **Reed-Solomon code**.* □

It is a well-known fact that $\mathrm{RS}(P, k)$ has length n, dimension k, and minimum distance $d = n - k + 1$. Furthermore, the weight distribution of Reed-Solomon codes is well-known (in fact, for given n, k, and q all MDS-codes have the same weight distribution). The following proposition can be found in various text-books, for example [2], Theorem 14.1.2.

Proposition 3.2 (Weight distribution of Reed-Solomon codes)
If

$$A_u = |\{c \in \mathrm{RS}(P, k) \mid \mathbf{w}(c) = u\}|$$

then $A_0 = 1$, $A_u = 0$ for $1 \leq u < d$ and

$$A_u = \binom{n}{u} \sum_{i=0}^{u-d} (-1)^i \binom{u}{i} (q^{u-d+1-i} - 1)$$

for $d \leq u \leq n$. □

Let $\mathrm{B}(w, r)$ denote the **ball** with radius r and center w. That is

$$\mathrm{B}(w, r) = \{u \in \mathbb{F}_q^n \mid \mathrm{d}(w, u) \leq r\}$$

The **sphere** with radius r and center w will be denoted by

$$\mathrm{S}(w,r) = \{u \in \mathbb{F}_q^n \mid \mathrm{d}(w,u) = r\}$$

Suppose that w is a received word and that $\mathrm{RS}(P,k)$ is the code in use. Then decoding up to τ errors from w can be specified as calculating the following set:

$$\mathrm{dec}_\tau(w) = \mathrm{B}(w,\tau) \cap \mathrm{RS}(P,k)$$

Notice that if $|\mathrm{dec}_\tau(w)| = 0$ then more than τ errors have been detected. If $\tau \leq \left\lfloor \frac{d-1}{2} \right\rfloor$ then for any w, $|\mathrm{dec}_\tau(w)| \leq 1$. If $\tau > \left\lfloor \frac{d-1}{2} \right\rfloor$ then decoding τ errors is often referred to as list-decoding, since $\mathrm{dec}_\tau(w)$ may be a list of several codewords, all within distance τ from the received word. In this text, the codewords in $\mathrm{dec}_\tau(w)$ will be called the **candidates** of decoding τ errors from w.

The decoding algorithm uses bivariate polynomials and we need an ordering of these.

Let $M = \{x^\alpha y^\beta \in \mathbb{F}_q[x,y] \mid (\alpha,\beta) \in \mathbb{N}^2\}$ be the set of monomials in $\mathbb{F}_q[x,y]$. A **monomial ordering** is a relation, $\leq_m$, on M, which satisfies the following:

- $\leq_m$ is total.
- $\forall f,g,h \in M : f \leq_m g \Rightarrow fh \leq_m gh$
- $\leq_m$ is a well-ordering.

One monomial ordering is the **lexicographic order** defined by

$$x^\alpha y^\beta \leq_l x^a y^b \Leftrightarrow \alpha < a \vee (\alpha = a \wedge \beta \leq b)$$

In the following this will be called the lexicographic order with $x < y$. In a similar way we may define a lexicographic order with $y < x$ by exchanging x and y in the expression above.

Let $f(x,y) \in \mathbb{F}_q[x,y] \backslash \{0\}$. Then the **weighted degree** of $f(x,y)$ is given by

$$\deg^{(a,b)}(f) = \max\{\alpha a + \beta b \mid f_{(\alpha,\beta)} \neq 0\}$$

where $a \in \mathbb{N}$ is called the weight of x and $b \in \mathbb{N}$ is called the weight of y. For any choice of a and b we may define $\deg^{(a,b)}(0) = -\infty$.

Given a weighted degree, $\deg^{(a,b)}$, and a lexicographic order, $\leq_l$, we can define a corresponding **weighted degree lexicographic order** on M by

$$f \leq_w g \Leftrightarrow \deg^{(a,b)}(f) < \deg^{(a,b)}(g) \vee (\deg^{(a,b)}(f) = \deg^{(a,b)}(g) \wedge f \leq_l g)$$

for all $f, g \in M$.

In the following, if $f \in \mathbb{F}_q[x, y]$, then $f_{(\alpha,\beta)}$ will be defined for any pair, $(\alpha, \beta) \in \mathbb{N}^2$ by

$$f = \sum_{(\alpha,\beta)\in\mathbb{N}^2} f_{(\alpha,\beta)} x^\alpha y^\beta$$

Furthermore, define

$$\operatorname{coef}(f, x^\alpha y^\beta) = f_{(\alpha,\beta)}$$

3.3 Sudan's algorithm

The following formulation of Sudan's algorithm is inspired by the presentation of Sudan's original algorithm by W. Feng and R. E. Blahut.

Algorithm 3.3 (Sudan's extended algorithm)
Input: The code $\mathrm{RS}(P, k)$*, a received word,* $w \in \mathbb{F}_q^n$*, and a parameter,* $s \geq 1$.
Output: $\operatorname{dec}_{\tau_s}(w)$.

- *Calculate* r_s *so that*

$$\binom{r_s}{2} \leq \frac{n\binom{s+1}{2}}{k-1} < \binom{r_s+1}{2} \tag{3.1}$$

and let

$$\ell_s = \left\lfloor \frac{n\binom{s+1}{2}}{r_s} + \frac{(r_s-1)(k-1)}{2} \right\rfloor$$

Then

$$\tau_s = n - \left\lfloor \frac{\ell_s}{s} \right\rfloor - 1 \tag{3.2}$$

- *Calculate* $Q_s(x, y) \in \mathbb{F}_q[x, y]\backslash\{0\}$ *so that*

1. $\mathrm{coef}(Q_s(x+P_i, y+w_i), x^\alpha y^\beta) = 0$ *for all* $\alpha, \beta \in \mathbb{N}$ *with* $\alpha+\beta < s$.
2. $\deg^{(1,k-1)}(Q_s) \leq \ell_s$

- *Factorize* Q_s *into irreducible factors.*
- *If* $y - f(x)$ *with* $\deg(f) < k$ *divides* Q_s *and* $\mathrm{d}(f(P), w) \leq \tau_s$ *then include* $f(P)$ *in the set of candidates. That is*

$$\mathrm{dec}_{\tau_s}(w) = \{f \in \mathbb{F}_q[x] \mid \deg(f) < k \wedge (y - f(x)) | Q_s \wedge \mathrm{d}(f(P), w) \leq \tau_s\}$$

□

If $s = 1$ then this algorithm is identical to Sudan's original algorithm [38].

Any non-zero polynomial satisfying conditions 1 and 2 on the polynomial Q_s will be called a Q_s-polynomial in the following.

Notice that condition 1 on a Q_s-polynomial states that (P_i, w_i) must be a zero of multiplicity s.

In the following it will be proven that Sudan's extended algorithm gives the promised result. This will be done by proving that a Q_s-polynomial exists and that it has the right factors of the form $y - f(x)$. Furthermore, it will be clear that a Q_s has at most $r_s - 1$ such factors, so the number of codewords in the output is upper bounded by $r_s - 1$.

Lemma 3.4 (Weighted degree of i$^{\text{th}}$ monomial)
Consider the polynomial ring $\mathbb{F}_q[x, y]$ *with the weighted degree* $\deg^{(1,k-1)}$ *and let the monomials be ordered by a corresponding* $\leq_w$ *order. Suppose that*

$$m_0 \leq_w m_1 \leq_w m_2 \leq_w \ldots$$

is an increasing list of all the monomials in $F_q[x, y]$. *Then*

$$\deg^{(1,k-1)}(m_i) = \left\lfloor \frac{i}{r} + \frac{(r-1)(k-1)}{2} \right\rfloor \tag{3.3}$$

where r *satisfies*

$$\binom{r}{2} \leq \frac{i}{k-1} < \binom{r+1}{2}$$

□

Proof:
Group the monomials into the disjoint sets, $M_1, M_2, \ldots$, where

$$M_c = \{m_j \mid (c-1)(k-1) \leq \deg^{(1,k-1)}(m_j) < c(k-1)\}$$

Then $|M_c| = c(k-1)$ so $|M_1| + |M_2| + \ldots + |M_{c-1}| = \binom{c}{2}(k-1)$. Since $\binom{r}{2}(k-1) \leq i < \binom{r+1}{2}(k-1)$, $m_i \in M_r$. The smallest monomial in M_r has weighted degree $(r-1)(k-1)$ and for each a with $(r-1)(k-1) \leq a < r(k-1)$ there are exactly r monomials with weighted degree a in M_r. If the monomials of M_r are listed increasingly with respect to $\leq_w$ then m_i, is monomial number $i - \binom{r}{2}(k-1)$, so the weighted degree of m_i must be

$$\deg^{(1,k-1)}(m_i) = (r-1)(k-1) + \left\lfloor \frac{i - \binom{r}{2}(k-1)}{r} \right\rfloor = \left\lfloor \frac{i}{r} + \frac{(r-1)(k-1)}{2} \right\rfloor$$

■

Lemma 3.5 (Transformation of polynomial)
Let $f(x,y) \in \mathbb{F}_q[x,y]$ and $x_0, y_0 \in \mathbb{F}_q$ then

$$\text{coef}(f(x+x_0, y+y_0), x^\alpha y^\beta) = \sum_{a \geq \alpha \wedge b \geq \beta} \binom{a}{\alpha}\binom{b}{\beta} \text{coef}(f(x,y), x^a y^b) x_0^{a-\alpha} y_0^{b-\beta}$$

for all $(\alpha, \beta) \in \mathbb{N}^2$. □

Proof:
This is seen by direct calculation. ■

Theorem 3.6 (Existence of Q_s-polynomial)
A Q_s-polynomial does exist. □

Proof:
Notice that by Lemma 3.5 the first condition on a Q_s-polynomial form $n\binom{s+1}{2}$ linear equations. Therefore, a Q_s-polynomial can be constructed by solving a system of $n\binom{s+1}{2}$ linear equations with $n\binom{s+1}{2}+1$ unknowns, where the unknowns are the coefficients of Q_s using the basis monomials $m_0, \ldots, m_n$ as defined in Lemma 3.4.

By Lemma 3.4 the weighted degree of Q_s will satisfy

$$\deg^{(1,k-1)}(Q_s) \leq \deg^{(1,k-1)}(m_{n\binom{s+1}{2}}) = \left\lfloor \frac{n\binom{s+1}{2}}{r_s} + \frac{(r_s-1)(k-1)}{2} \right\rfloor = \ell_s$$

■

Lemma 3.7 (Factors of $Q_s(x, f(x))$)
Let $f(x)$ be a polynomial and suppose that $f(P_i) = w_i$. Then

$$(x - P_i)^s | Q_s(x, f(x))$$

□

Proof:
Let $p(x) = f(x + P_i) - w_i$. Then $p(0) = 0$ so $x|p(x)$. Now consider $h(x) = Q_s(x + P_i, p(x) + w_i)$. By the definition of Q_s, $h(x)$ has no terms of degree less than s (due to the fact that x divides $p(x)$). So $x^s|h(x)$ and $(x - P_i)^s|h(x - P_i)$. This proves the lemma since $h(x - P_i) = Q_s(x, f(x))$. ■

Theorem 3.8 (Factors of the Q_s-polynomial)
Let $Q_s(x, y)$ be a Q_s-polynomial corresponding to the received word, w. If $f \in \mathbb{F}_q[x]$ with $\deg(f) < k$ satisfies $\mathrm{d}(f(P), w) \leq \tau_s$ then $(y - f(x))|Q_s(x, y)$.

□

Proof:
Let $g(x) = Q_s(x, f(x))$. Then $\deg(g) \leq \ell_s$ and by Lemma 3.7 $(x - P_i)^s|g(x)$ for each value of i where $f(P_i) = w_i$. This happens at least $n - \tau_s$ times. This means that a polynomial of degree at least

$$s(n - \tau_s) = s(n - (n - \left\lfloor \frac{\ell_s}{s} \right\rfloor - 1)) = s\left(\left\lfloor \frac{\ell_s}{s} \right\rfloor + 1\right) > \ell_s$$

divides $g(x)$. So $g(x)$ must be the 0-polynomial.

Now consider $Q_s(x, y)$ as a polynomial in y where the coefficients are polynomials in x. As shown, $f(x)$ is a zero of this polynomial, so $y - f(x)$ must divide $Q_s(x, y)$. ■

3.4 Error-correcting capability of Sudan's algorithm

The error-correcting capability, τ_s, of Sudan's algorithm depends on the code rate, $\frac{k}{n}$, and on the parameter, s. Asymptotically, we have the following result:

Theorem 3.9 (Asymptotic error-correcting capability)
Suppose that n and s are "big" and that $k = \rho n$. Then

$$\frac{\tau_s}{n} \approx 1 - \sqrt{\rho}$$

□

Proof:
By the definition of τ_s (Equation 3.2) we see that

$$\frac{\tau_s}{n} \approx 1 - \frac{\ell_s}{ns}$$

The definition of r_s (Equation 3.1) gives the following approximation:

$$\frac{r_s^2}{s^2} \approx \frac{1}{\rho}$$

Then look at $\frac{\ell_s}{ns}$:

$$\frac{\ell_s}{ns} \approx \frac{s}{2r_s} + \rho\frac{r_s}{2s} \approx \frac{1}{2}\sqrt{\rho} + \frac{1}{2}\sqrt{\rho} = \sqrt{\rho}$$

■

The asymptotic error-correcting capability obtained by Sudan's extended algorithm can actually be shown to be the best possible for a list-decoder where the size of the list of candidates is bounded independently of the code length [19].

Figure 3.1 shows the actual error-correcting capability of Sudan's extended algorithm for $n = q = 64$ and $s \in \{1, 4\}$ together with the asymptotical error-correcting capability.

3.5 The number of candidates

The next theorem shows that the number of candidates does not increase with the code length nor with the size of the alphabet:

Theorem 3.10 (Upper bound on number of candidates)
For any $w \in \mathbb{F}_q^n$, $|\mathrm{dec}_{\tau_s}(w)| < r_s$. □

Proof:
As seen in the proof of Lemma 3.4, M_{r_s} only contains monomials with degree in y at most $r_s - 1$ so $\deg^{(0,1)}(Q_s(x, y)) < r_s$. Therefore, Q_s can have at most $r_s - 1$ factors of the form $y - f(x)$. This proves the theorem.

■

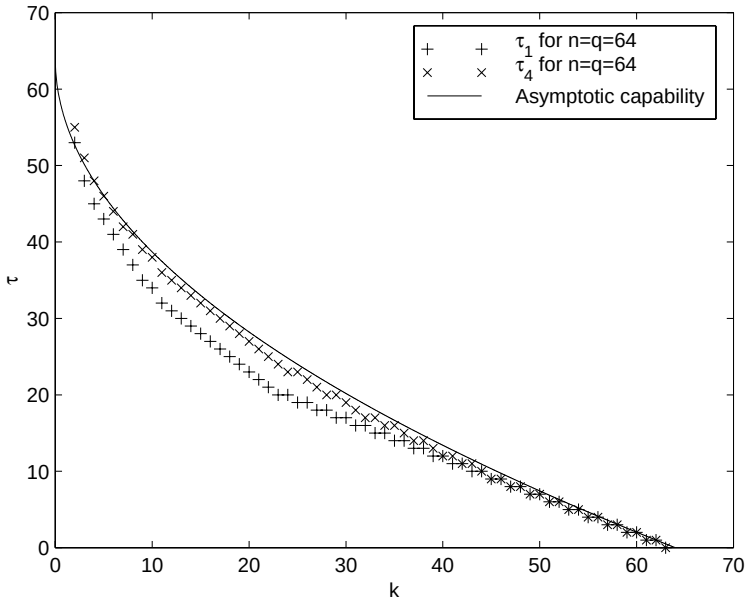

Figure 3.1: *Error-correcting capability of Sudan's extended algorithm.*

The occurrence of various candidates is a problem in applications, where one would like to end up with a unique answer. In the following we will analyse (under certain assumptions) how often the algorithm actually will return more than one candidate. Similar results can be found in [2], Sect. 14.2.

Suppose that all error vectors with weight at most τ occur with the same probability regardless of the weight of the vector. Let v be some transmitted codeword and assume that at most τ errors occurred. If w is the received word then $\mathrm{P}(|\mathrm{dec}_\tau(w)| > 1)$ will be given by the fraction of words in $\mathrm{B}(v, \tau)$ which are also within distance τ from another codeword. Since the code is linear it will be sufficient to analyse the situation in the case $v = 0$. Unfortunately, an exact count of the number of words in $\mathrm{B}(0, \tau)$ within distance τ from another codeword is very difficult if not impossible to make, but an upper bound is given by

$$M(\tau) = \sum_{c \in \mathrm{RS}(P,k) \setminus \{0\}} |\mathrm{B}(0, \tau) \cap \mathrm{B}(c, \tau)|$$

(equality holds when $r_s \leq 3$ in Sudan's algorithm).

In the following an explicit expression for $M(\tau)$ will be found. The first step is taken by this lemma:

Lemma 3.11 (Intersection of spheres)

Let $c \in \mathrm{RS}(P,k)$ *and let* $i, a < \mathbf{w}(c)$. *If* $\mathbf{w}(c) - a > i$ *then*

$$|\mathrm{S}(0,a) \cap \mathrm{S}(c,i)| = 0$$

If $\mathbf{w}(c) - a \leq i$ *then*

$$|\mathrm{S}(0,a) \cap \mathrm{S}(c,i)| = \sum_{j=0}^{\ell} \binom{u}{u-a+j} \binom{n-u}{j} (q-1)^j \binom{a-j}{i-(u-a)-2j} (q-2)^{i-(u-a)-2j}$$

where $u = \mathbf{w}(c)$ *and*

$$\ell = \min\left\{ \left\lfloor \frac{i-(u-a)}{2} \right\rfloor, n-u \right\}$$

□

Proof:
The problem is to calculate the number of words with weight a which have distance i to c, or put in another way: in how many ways c can be changed on exactly i positions to get a word with weight a.

Suppose that exactly j positions where c was previously 0 are now changed to a non-zero value. Then $j \leq n - \mathbf{w}(c)$ which is the number of zeroes and $j \leq \left\lfloor \frac{i-(\mathbf{w}(c)-a)}{2} \right\rfloor$, because to end at the right weight, a, it will be necessary to change $\mathbf{w}(c) - a + j$ of the previously non-zero values to zero. The number of ways to choose the $\mathbf{w}(c) - a + j$ "non-zero to zero" positions will be $\binom{\mathbf{w}(c)}{\mathbf{w}(c)-a+j}$. The number of ways to choose the j "zero to non-zero" positions will be $\binom{n-\mathbf{w}(c)}{j}$ and each position can be given any of the $q-1$ non-zero values. What is left to change $i - (\mathbf{w}(c) - a + j) - j$ of the remaining non-zero values to another non-zero value. The positions can be chosen in $\binom{a-j}{i-(\mathbf{w}(c)-a)-2j}$ ways and each position can be given any of the $q-2$ other non-zero values.

This gives the expression in the lemma. ■

Now $M(\tau)$ can be calculated as

$$M(\tau) = \sum_{u=d}^{\min\{2\tau,n\}} A_u \sum_{a=u-\tau}^{\tau} \sum_{i=u-a}^{\tau} |\mathrm{S}(0,a) \cap \mathrm{S}(c,i)| \tag{3.4}$$

where A_u is given by Proposition 3.2.

This gives an upper bound for the wanted probability:

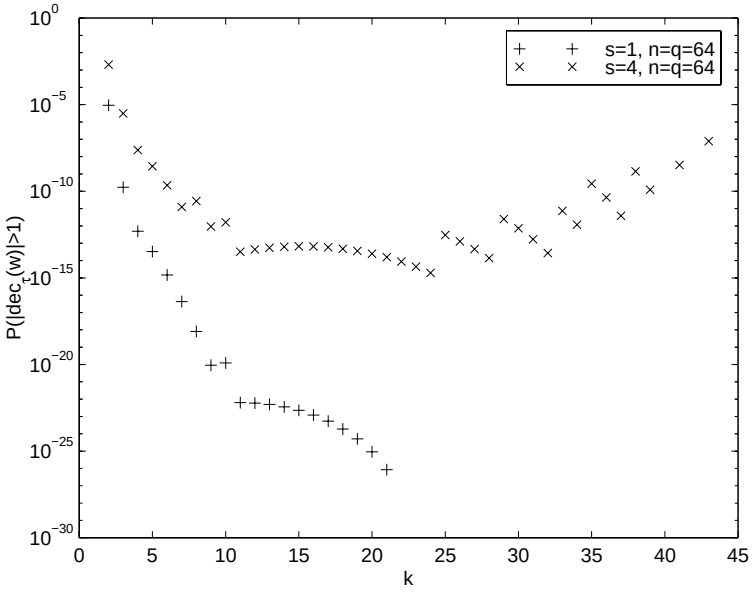

Figure 3.2: *Upper bound on probability of getting more than one candidate as output from Sudan's extended algorithm when* $n = q = 64$ *and* $s \in \{1, 4\}$. *The bound is zero when* $\tau_s = \left\lfloor \frac{d_{\min}-1}{2} \right\rfloor$.

Proposition 3.12 (Probability of multiple candidates)
The probability that Sudan's algorithm returns more than 1 *codeword given that at most* τ *errors have occurred (assuming that all error-patterns occur with the same probability, regardless of the weight) satisfies*

$$\mathrm{P}(|\mathrm{dec}_\tau(w)| > 1) \;=\; \frac{M(\tau)}{|\mathrm{B}(0,\tau)|}$$

□

Proof:
This is a direct consequence of the above calculations. ■

Due to the complexity even of the calculation of the upper bound, $\frac{M(\tau)}{|\mathrm{B}(0,\tau)|}$ it is difficult to get an immediate estimate of this probability. However, it turns out that it is usually very small. For example, Fig. 3.2 shows the upper bound for $n = q = 64$ and $s \in \{1, 4\}$. This figure should be compared to Fig. 3.1, which shows the corresponding error-correcting capability.

3.6 Determining a Q_s-polynomial

Algorithm 3.3 contains two main steps. The first is to determine a Q_s-polynomial and the second is to identify factors of the form $y - f(x)$ with

$\deg(f) < k$. This section and the next section describe how to make an efficient implementation of these two steps.

To determine the Q_s-polynomial we must find a polynomial with lowest possible weighted degree which has each point (P_i, w_i) as a zero of multiplicity s. The straight-forward way to do this is to solve a system of $\binom{s+1}{2}n$ equations. However, in the following we present a faster algorithm, which is an application of the Fundamental Iterative Algorithm, along the same lines as the application in [20], Chap. 4. The Fundamental Iterative Algorithm was first presented in [7].

Let the monomials of $\mathbb{F}_q[x, y]$ be $m_0, m_1, \ldots$ as in Lemma 3.4. Now define $\text{ord} : \mathbb{F}_q[x, y] \to \mathbb{N} \cup \{-\infty\}$ by $\text{ord}(0) = -\infty$, $\text{ord}(\text{span}\{m_0\}\backslash\{0\}) = \{0\}$, and

$$\text{ord}(\text{span}\{m_0, \ldots, m_i\}\backslash\text{span}\{m_0, \ldots, m_{i-1}\}) = \{i\}$$

for $i > 0$. Notice that ord induces both an partitioning of the polynomials in $\mathbb{F}_q[x, y]$ and a total ordering on the set of partitions. As it will be seen later, the polynomials which occur at a given stage in the algorithm all come from different partitions. It will therefore be well-defined to talk about the smallest or the minimal polynomial as the polynomial which belongs to the smallest partition.

Let $g_1, \ldots, g_c \in \mathbb{F}_q[x, y]$ be a set of polynomials so that if

$$A_j = \text{ord}(\{x^{is} g_j \mid i \in \mathbb{N}\}) \ , \quad j = 1, \ldots, c$$

then $A_j \cap A_{j'} = \emptyset$ for $j \neq j'$ and $\bigcup_{j=1}^{c} A_j = \mathbb{N}$.

Furthermore, define

$$G_j = \{g \in \mathbb{F}_q[x, y] \mid \text{ord}(g) \in A_j\}$$

Finally, let

$$M_s = \{x^\alpha y^\beta \mid \alpha + \beta < s\} = \{e_1, \ldots, e_{|M_s|}\}$$

where $|M_s| = \binom{s+1}{2}$.

The following is a mathematical description of the interpolation algorithm. This will be used to show the correctness of the algorithm. A more implementation-oriented description is given in Algorithm 3.15.

First, let

$$G^{(0)} = \{{g_1}^{(0)}, \ldots, {g_c}^{(0)}\}$$

where ${g_j}^{(0)} = g_j$.

The sets $G^{(1)}, \ldots, G^{(n)}$ are then constructed iteratively using the following method:

Let $G^{(i,0)} = G^{(i-1)}$ and do the following for each $k = 1, \ldots, |M_s|$:

Let $G^\star = \{g \in G^{(i,k-1)} \mid \mathrm{coef}(g(x + x_i, y + y_i), e_k) \neq 0\}$. If $G^\star = \emptyset$ then $G^{(i,k)} = G^{(i,k-1)}$. If not, let $f^{(i,k)}$ be chosen so that

$$\mathrm{ord}(f^{(i,k)}) = \min \mathrm{ord}(G^\star)$$

Now let

$$\begin{aligned} G^{(i,k)} = \{(x - x_i)^s f^{(i,k)}\} \cup \{&\mathrm{coef}(f^{(i,k)}(x + x_i, y + y_i), e_k) g - \\ &\mathrm{coef}(g(x + x_i, y + y_i), e_k) f^{(i,k)} \mid g \in G^{(i,k-1)} \backslash \{f^{(i,k)}\}\} \end{aligned}$$

where $p_i = (x_i, y_i)$.

Let $G^{(i)} = G^{(i,|M_s|)}$.

The result is now given by the polynomial in $G^{(n)}$ which is smallest with respect to ord.

To prove the correctness of the algorithm, the following lemma is needed:

Lemma 3.13 (Correctness of one step of the interpolation algorithm)
If ${g_j}^{(i,k-1)}$ *is a minimal polynomial with respect to* ord *in* G_j *where*

$$\mathrm{coef}({g_j}^{(i,k-1)}(x + x_{i'}, y + y_{i'}), e_{k'}) = 0$$

for all $i' \leq i$ *and* $k' \leq k - 1$ *then the similar condition is true for* ${g_j}^{(i,k)}$. □

Proof:
Assume that ${g_j}^{(i,k-1)}$ is a minimal polynomial with respect to ord in G_j where $\mathrm{coef}({g_j}^{(i,k-1)}(x + x_{i'}, y + y_{i'}), e_{k'}) = 0$ for all $i' \leq i$ and $k' \leq k - 1$.

If ${g_j}^{(i,k-1)} \neq f^{(i,k)}$ then $\mathrm{ord}({g_j}^{(i,k)}) = \mathrm{ord}({g_j}^{(i,k-1)})$ and the lemma is true by assumption.

If ${g_j}^{(i,k-1)} = f^{(i,k)}$ then let g be a polynomial with $\mathrm{ord}(g) = \mathrm{ord}({g_j}^{(i,k-1)})$ and $\mathrm{coef}(g(x + x_{i'}, y + y_{i'}), e_{k'}) = 0$ for $k' \leq k$. It may be assumed that both g and ${g_j}^{(i,k-1)}$ are monic. Now consider the polynomial $h = {g_j}^{(i,k-1)} - g$. $\mathrm{ord}(h) < \mathrm{ord}({g_j}^{(i,k-1)})$ but $\mathrm{coef}(h(x + x_i, y + y_i), e_k) \neq 0$ so $h \in G_j$ since the definition of $f^{(i,k)}$ implies that it cannot be in any $G_{j'}$ for $j' \neq j$. However, this contradicts the assumption that ${g_j}^{(i,k-1)}$ is minimal. So a

polynomial like g does not exist. Therefore it is necessary that $\mathrm{ord}({g_j}^{(i,k)}) > \mathrm{ord}({g_j}^{(i,k-1)})$ and since ${g_j}^{(i,k)} = (x - x_i)^s {g_j}^{(i,k-1)}$ introduces the smallest possible increase with respect to ord, the theorem is also true for ${g_j}^{(i,k)}$. ■

The correctness of the algorithm follows from this theorem:

Theorem 3.14 (Correctness of the interpolation algorithm)
For each $j \in \{1, \ldots, c\}$ and each $i = 0, \ldots, n$, ${g_j}^{(i)}$ is a minimal polynomial with respect to ord *in G_j where* $\mathrm{coef}({g_j}^{(i)}(x + x_{i'}, y + y_{i'}), e_{k'}) = 0$ *for all $i' \leq i$ and $k' \leq |M_s|$.* □

Proof:
The proof is by induction over i, however, most of the work has been done in Lemma 3.13.

$i' = 0$: The polynomials ${g_{i'}}^{(0)}$ are minimal in the sets $G_{i'}$ by construction so the theorem is true for $i' = 0$.

$i' = i + 1$: Assume that the theorem is true for i. Since ${G_j}^{(i',0)} = {G_j}^{(i,|M_s|)}$ it follows by Lemma 3.13 and induction over k that the theorem is true for i'.

■

The following shows how to implement the interpolation algorithm:

Algorithm 3.15 (Fast interpolation)
Input: Interpolation points, $p = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, a polynomial ordering, ord*, and a required root multiplicity, s.*
Output: Interpolation polynomial which is minimal with respect to ord*.*

$Result = Interpolate(p, \mathrm{ord}, s)$:
$d \leftarrow \max\{\deg^{(0,1)} m_i \mid i = 0, \ldots, n\binom{s+1}{2}\}$
$c \leftarrow 0$
for $j' = 0, \ldots, d$ **do**
  for $k = 1, \ldots, s$ **do**
    $c \leftarrow c + 1$
    $g_c \leftarrow x^{k-1} y^j$
  end
end
for $i = 1, \ldots, n$ **do**

for $k = 1, \ldots, |M_s|$ **do**
$f \leftarrow \min_{\text{ord}}\{g_j \mid \text{coef}(g_j(x + x_i, y + y_i), e_k) \neq 0\}$
for $j = 1, \ldots, c$ **do**
if $g_j = f$ **then**
$g_j \leftarrow (x - x_i)^s g_j$
else
$g_j \leftarrow \text{coef}(f(x+x_i, y+y_i), e_k)g_j - \text{coef}(g_j(x+x_i, y+y_i), e_k)f$
end
end
end
end
$Result \leftarrow \min_{\text{ord}}\{g_j\}$ □

Notice that $\text{coef}(f(x+x_0, y+y_0), e)$ for some polynomial, f, some scalars, x_0 and y_0, and some monomial, e, can be calculated using the result in Lemma 3.5.

3.7 Factoring the Q_s-polynomial

Q_s is a bivariate polynomial, but the goal is to identify factors of the form $y - f(x)$ with $\deg(f) < k$. Therefore, let $E = \mathbb{F}_{q^k} = \mathbb{F}_q[x]/\langle e(x)\rangle$ where $e(x)$ is an irreducible polynomial in $\mathbb{F}_q[x]$ of degree k. E is then a finite field with q^k elements.

Consider the map $\phi : \mathbb{F}_q[x, y] \to E[y]$ given by

$$\phi(\sum_i p_i(x)y^i) = \sum_i [p_i(x)]_E \, y^i \tag{3.5}$$

It is a well-known fact that

Lemma 3.16
ϕ is a ring homomorphism. □

Theorem 3.17
If $f(x, y)|Q(x, y)$ then $\phi(f)|\phi(Q)$. □

Proof:
The theorem follows from Lemma 3.16:

$$f|Q \Rightarrow Q = fg \Rightarrow \phi(Q) = \phi(fg) = \phi(f)\phi(g) \Rightarrow \phi(f)|\phi(Q)$$

for some $g \in F_q[x, y]$. ■

Corollary 3.18
If $(y - f(x))|Q(x, y)$ then $y - [f]_E$ is an irreducible factor of $\phi(Q)$. □

Proof:
If $(y - f(x))|Q(x, y)$ then $y - [f]_E$ is a factor of Q by the theorem. Furthermore, $y - [f]_E$ must be irreducible since it is a polynomial of degree 1. ■

Theorem 3.17 reduces the problem of factoring the bivariate polynomial Q_s into the problem of factoring the univariate polynomial $\phi(Q_s)$, which is much easier.

The following is a short description of Berlekamp's algorithm for factoring univariate polynomials over a large finite field. This version of the algorithm is from [12], which describes more details and proves the correctness of the algorithm.

First the factoring problem is reduced to factoring monic square-free polynomials by the following algorithm:

Algorithm 3.19 (Square-free factorization)
Input: A monic polynomial, $a(x) \in \mathbb{F}_{p^m}[x]$.
Output: $a(x)$ factorized on the form $a = \prod_j a_{[j]}^j$ where each $a_{[j]} \in \mathbb{F}_{p^m}[x]$ is square-free.

$Result = SquareFree(a)$:
$j \leftarrow 1,\ Result \leftarrow 1$
if $a' = 0$ **then**
  $Result \leftarrow Result \cdot SquareFree(a^{1/p})^p$
else
  $\mathsf{c} \leftarrow \gcd(\mathsf{a}, \mathsf{a}')$ — c holds the repeated factors.
  $\mathsf{w} \leftarrow \mathsf{a}/\mathsf{c}$ — w is a square-free polynomial containing each of the factors except those where p divides the exponent.

while $w \neq 1$ **do**

$\mathsf{y} \leftarrow \gcd(\mathsf{w}, \mathsf{c})$	y becomes a square-free polynomial containing the repeated factors.
$\mathsf{z} \leftarrow \mathsf{w}/\mathsf{y}$	Now z contains all the non-repeated factors ($a_{[j]} = z$).
$Result \leftarrow Result \cdot (z^j),\ j \leftarrow j + 1$	
$\mathsf{w} \leftarrow \mathsf{y}$	Remove the factors which are now contained in the result.
$\mathsf{c} \leftarrow \mathsf{c}/\mathsf{y}$	Decrease the exponent of each factor by 1 except where p divides the exponent.

if $c \neq 1$ **then**

$Result \leftarrow Result \cdot SquareFree(c^{1/p})^p$

end

end □

In the *while*-loop in the algorithm, factors with exponent divisible by p will survive in the polynomial c so that when all the factors of w has been included in the result, c is either 1 or all its factors has an exponent divisible by p.

If $a(x) \in \mathbb{F}_q[x]$ is a monic square-free polynomial then we will define the vector space

$$W(a) = \{v \in \mathbb{F}_q[x]/\langle a(x)\rangle \mid v^q = v\}$$

The following algorithm for factoring a monic square-free polynomial assumes the presence of a function, $WBasis$, which calculates a basis of $W(a)$.

Algorithm 3.20 (Berlekamp's factoring algorithm)
Input: A monic square-free polynomial, $a(x) \in \mathbb{F}_{p^m}[x]$.
Output: The set of irreducible factors of $a(x)$.

$Result = Factor(a)$:

$Result \leftarrow \{a\}$

$(\mathsf{v}_1, \ldots, \mathsf{v}_k) \leftarrow \mathsf{WBasis(a)}$	Calculate k polynomials spanning W.

while $|Result| < k$ **do**
  for $u \in Result$ **do**
    $\mathsf{v} \leftarrow \mathsf{Random}(\mathsf{v}_1, \ldots, \mathsf{v}_k)$ Get a random polynomial from W.
    $v \leftarrow v + v^2 + \cdots + v^{2^{m-1}}$
    $g \leftarrow \gcd(v, u)$
    if $g \neq 1$ **and** $g \neq u$ **then**
      $Result \leftarrow Result \backslash \{u\}$
      $Result \leftarrow Result \cup \{\frac{u}{g}, g\}$
      if $|Result| = k$ **then return** $Result$
    end
  end
end □

This factoring algorithm is probabilistic and requires $O(r^2kt\log(t) + r^3)$ operations in $\mathbb{F}_{q^k}$. Here, t is the number of irreducible factors, r is the degree of the polynomial, and k is the logarithm of the field size.

Another efficient method for finding factors of the form $y - f(x)$ is given in [11].

3.8 Examples

This section contains two examples of decoding by Sudan's extended algorithm. The code alphabet is $\mathbb{F}_{16}$ with primitive element, α, satisfying $\alpha^4 + \alpha + 1 = 0$. The Reed-Solomon code is $\mathrm{RS}(\{1, \alpha, \alpha^2, \ldots, \alpha^{14}\}, 7)$ so $(n, k, d) = (15, 7, 9)$. This gives $\tau_4 = 5$ which is one more than half the minimum distance.

In the first example, $c = (0, \ldots, 0)$ is the transmitted codeword, and we generate 5 random errors, so the received word is

$$w = (0, 0, \alpha^{11}, 0, \alpha^{12}, \alpha^{11}, 0, 0, 0, 0, 0, 0, \alpha^3, 0, \alpha^7)$$

The first step of Sudan's extended algorithm with $s = 4$ gives the following

interpolation polynomial:

$$\begin{aligned}Q_4 = {} & (\alpha^5 + \alpha^4 x + \alpha^2 x^2 + x^3 + \alpha^7 x^4 + \alpha^3 x^5 + \alpha^6 x^6 + \alpha^3 x^7 + \alpha^7 x^8 + \alpha^6 x^9 + \\ & \alpha^2 x^{10} + \alpha^{13} x^{11} + \alpha^9 x^{12} + \alpha^{10} x^{13} + \alpha^3 x^{14} + \alpha^{11} x^{15} + x^{17} + \alpha^{14} x^{18} + \\ & \alpha^2 x^{19} + \alpha^4 x^{20} + \alpha^{10} x^{21} + \alpha^{12} x^{22} + \alpha^6 x^{24} + \alpha^2 x^{25} + \alpha^2 x^{26} + \alpha x^{27} + \\ & \alpha^3 x^{28} + \alpha^8 x^{29} + \alpha^9 x^{31} + \alpha^{13} x^{32} + x^{33}) y + (\alpha^4 x + x^2 + \alpha^{11} x^3 + \alpha^8 x^4 + \\ & x^5 + \alpha^{14} x^6 + \alpha^3 x^7 + \alpha^{14} x^8 + \alpha^5 x^9 + \alpha^7 x^{10} + \alpha^{11} x^{11} + \alpha^6 x^{12} + \alpha x^{13} + \\ & \alpha^9 x^{14} + \alpha^{11} x^{15} + \alpha x^{16} + \alpha^{13} x^{17} + \alpha^{11} x^{18} + \alpha^{11} x^{19} + \alpha^{14} x^{20} + \alpha^{10} x^{21} + \\ & \alpha^5 x^{22} + \alpha^{13} x^{23} + \alpha x^{24} + \alpha^4 x^{25} + \alpha^3 x^{26} + \alpha^9 x^{27}) y^2 + (\alpha^9 + \alpha^5 x + \\ & \alpha^9 x^2 + \alpha^3 x^3 + \alpha^{10} x^4 + \alpha^4 x^6 + \alpha^9 x^{15} + \alpha^5 x^{16} + \alpha^9 x^{17} + \alpha^3 x^{18} + \\ & \alpha^{10} x^{19} + \alpha^4 x^{21}) y^3 + (\alpha^2 + \alpha^3 x + \alpha^5 x^2 + \alpha^{10} x^3 + \alpha x^4 + \alpha^{10} x^5 + \\ & \alpha^{14} x^7 + \alpha^4 x^8 + \alpha^5 x^9 + \alpha^{12} x^{10} + \alpha^{12} x^{11} + \alpha^{12} x^{12} + x^{13} + x^{14}) y^4 + \\ & (\alpha^{12} + \alpha^{14} x + \alpha^{13} x^2 + \alpha^4 x^3 + \alpha^{10} x^4 + \alpha^9 x^5 + \alpha^{11} x^6 + \\ & \alpha^4 x^7 + \alpha^8 x^8) y^5 + (\alpha x + \alpha^8 x^2) y^6\end{aligned}$$

This is reduced modulo $e(x) = 1 + \alpha^{11} x + \alpha^6 x^2 + \alpha^8 x^3 + \alpha^{11} x^4 + \alpha^{12} x^5 + \alpha^{13} x^6 + x^7$ which is irreducible over $\mathbb{F}_{16}$:

$$\begin{aligned}\phi(Q_4) = {} & (\alpha^8 + \alpha^{14} x + \alpha^9 x^2 + \alpha^{10} x^3 + x^4 + \alpha^5 x^5 + \alpha^{12} x^6) y + (\alpha^8 + \alpha x + \\ & \alpha^{11} x^2 + \alpha^4 x^3 + \alpha^7 x^4 + \alpha^6 x^5 + x^6) y^2 + (\alpha^8 + \alpha^{10} x + \alpha^8 x^2 + x^3 + \\ & \alpha^2 x^4 + x^5) y^3 + (\alpha^4 + \alpha^{14} x + \alpha^8 x^2 + \alpha^{11} x^3 + \alpha^9 x^4 + \alpha^9 x^5 + \\ & \alpha x^6) y^4 + (\alpha^5 + \alpha^2 x + x^2 + \alpha^9 x^3 + \alpha x^4 + \alpha^3 x^5 + x^6) y^5 + y^6\end{aligned}$$

Factoring this into irreducible factors over $\mathbb{F}_{16^7} = \mathbb{F}_{16}/\langle e(x) \rangle$ gives:

$$\begin{aligned}\phi(Q_4) = {} & y \cdot \\ & ((\alpha^4 + \alpha^{10} x + \alpha^{11} x^2 + \alpha^9 x^3 + \alpha^4 x^4 + \alpha^9 x^5 + x^6) + y) \cdot \\ & ((1 + \alpha^{14} x + \alpha^3 x^2 + \alpha^5 x^3 + \alpha^7 x^4 + \alpha^3 x^5 + \alpha x^6) + (\alpha^{13} + \alpha^{10} x + \\ & \alpha^7 x^2 + \alpha x^3 + \alpha^{10} x^4 + \alpha^7 x^5 + \alpha^4 x^6) y + (\alpha^8 + \alpha^8 x + \alpha^{12} x^2 + \\ & \alpha^3 x^3 + \alpha^{13} x^4 + \alpha^6 x^6) y^2 + (\alpha^8 + \alpha^4 x + \alpha^{12} x^2 + x^4 + \alpha x^5) y^3 + y^4)\end{aligned}$$

The factor $(\alpha^4 + \alpha^{10} x + \alpha^{11} x^2 + \alpha^9 x^3 + \alpha^4 x^4 + \alpha^9 x^5 + x^6) + y$ corresponds to a word with distance 14 to w. The only other factor of degree 1 is y. This gives

$$\mathrm{dec}_{\tau_4}(w) = \{(0, \ldots, 0)\}$$

which was the transmitted codeword. So in this case the decoding was not only correct, but also unique. As mentioned in Section 5 this is the normal case.

The next example was suggested by J. Justesen. The code is the same as before and the transmitted codeword is again the 0-word. However, the received word is now

$$v = (1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0)$$

Again, we let $s = 4$. The interpolation polynomial is

$$R = (\alpha^{14} + \alpha^{14}x^{10} + \alpha^{14}x^{20})y^2 + (\alpha^{14} + \alpha^{14}x^{10})y^4 + \alpha^{14}y^6$$

which is reduced modulo $e(x)$ to

$$\begin{aligned}\phi(R) = {} & (\alpha^{11} + \alpha^6x^2 + \alpha^2x^3 + \alpha^{10}x^4 + \alpha^8x^5 + \alpha^2x^6)y^2 + (\alpha^8 + \alpha^6x + \alpha^5x^2 + \\ & \alpha^8x^4 + \alpha^{10}x^5 + x^6)y^4 + y^6\end{aligned}$$

Factoring this polynomial into irreducibles gives

$$\phi(R) = ((\alpha^5 + \alpha^{10}x^5) + y)^2((\alpha^{10} + \alpha^5x^5) + y)^2y^2$$

Which corresponds to 3 codewords all with distance 5 to v:

$$\begin{aligned}\mathrm{dec}_{\tau_4}(v) = {} & \{(0, \ldots, 0), \\ & (1, \alpha^{10}, 0, 1, \alpha^{10}, 0, 1, \alpha^{10}, 0, 1, \alpha^{10}, 0, 1, \alpha^{10}, 0), \\ & (1, 0, \alpha^5, 1, 0, \alpha^5, 1, 0, \alpha^5, 1, 0, \alpha^5, 1, 0, \alpha^5)\}\end{aligned}$$

3.9 Conclusion

We have presented an efficient version of Sudan's algorithm for decoding Reed-Solomon codes. The corresponding algorithm for decoding algebraic geometry codes is the subject of a forthcoming paper.

Chapter 4

Decoding Hermitian codes with Sudan's algorithm

T. Høholdt and R. Refslund Nielsen[1]

Abstract

We present an efficient implementation of Sudan's algorithm for list decoding Hermitian codes beyond half the minimum distance. The main ingredients are a fast way of calculating so-called increasing zero bases, an efficient interpolation algorithm for finding the Q-polynomial, and a reduction of the problem of factoring the Q-polynomial to the problem of factoring a univariate polynomial over a large finite field.

4.1 Introduction

In 1997 M. Sudan [38] presented a conceptually easy algorithm for decoding Reed-Solomon codes of rates less than $\frac{1}{3}$ beyond half the minimum distance. The method was extended to algebraic geometry codes by M.A. Shokrollahi and H. Wassermann in [35], and M. Sudan and V. Guruswami in [13] further improved the algorithm to cover all rates both for Reed-Solomon codes and for general algebraic geometry codes. It is clear from [13] that the resulting algorithm has polynomial complexity, but also that some more work was needed to make the computations really efficient. In this paper we address that problem. The paper is organised as follows. Section 2 gives the necessary background on algebraic geometry codes, in particular the codes that come from the Hermitian curve. Section 3 gives the prerequisites

[1]Technical University of Denmark
Department of Mathematics, Bldg 303, DK-2800 Lyngby, Denmark. E-mails: tom@mat.dtu.dk and refslund@mat.dtu.dk

on multiplicities and the concept of increasing zero bases, and Section 4 presents and proves Sudan's algorithm. Section 5 gives an efficient method for calculating increasing zero bases and Section 6 gives a fast interpolation algorithm for finding the Q-polynomial. Section 7 treats the factorization problem and reduces it to factoring a univariate polynomial over a large finite field for which Berlekamp's algorithm [22] can be used. Section 8 contains an example and Section 9 is the conclusion.

4.2 Hermitian codes

Let χ be a nonsingular absolutely irreducible curve over $\mathbb{F}_q$ and let

$$P_1, \ldots, P_n, P_\infty$$

be $\mathbb{F}_q$-rational points on χ. The curve defines an algebraic function field, $\mathbb{F}_q(\chi)$, with a discrete valuation, $\mathbf{v}_{P_i}$, corresponding to each point ($i = 1, \ldots, n, \infty$).

Recall that a function, $f \in \mathbb{F}_q(\chi)$ is called regular in the point P_i if $\mathbf{v}_{P_i}(f) \geq 0$. The functions which are regular in a point form a ring, $\mathcal{O}_{P_i}$, which has a unique maximal principal ideal:

$$\mathcal{M}_{P_i} = \{f \in \mathbb{F}_q(\chi) \mid \mathbf{v}_{P_i}(f) > 0\} = \langle t_i \rangle$$

where t_i satisfies $\mathbf{v}_{P_i}(t_i) = 1$. t_i is then called a local parameter in P_i. Furthermore, the group of units of $\mathcal{O}_{P_i}$ is given by

$$\mathcal{O}_{P_i} \backslash \mathcal{M}_{P_i} = \{f \in \mathbb{F}_q(\chi) \mid \mathbf{v}_{P_i}(f) = 0\}$$

Any non-zero function, $f \in \mathbb{F}_q(\chi)$, can be written uniquely up to the choice of local parameter as follows:

$$f = t_i^{\mathbf{v}_{P_i}(f)} u_f$$

where u_f is a unit, that is $u_f \in \mathcal{O}_{P_i} \backslash \mathcal{M}_{P_i}$. This will be called the standard representation of f (with respect to the local parameter t_i). More details can be found in [37].

A class of algebraic geometry codes is given by

$$C_{\mathcal{L}}(P_1 + \ldots + P_n, mP_\infty) = \{(f(P_1), \ldots, f(P_n)) \mid f \in \mathcal{L}(mP_\infty)\} \quad , \; m < n$$

where

$$\mathcal{L}(mP_\infty) = \{f \in \mathbb{F}_q(\chi) \mid \mathbf{v}_{P_\infty}(f^{-1}) \leq m \wedge \mathbf{v}_{P_i}(f) \geq 0 \text{ for } i = 1, \ldots, n\}$$

The length of this code is n, and if g denotes the genus of χ and $2g-1 \leq m < n$ then the dimension of the code is $k = m - g + 1$ and the minimum distance is lower bounded by $d^* = n - m$ since the number of zeroes of a non-zero function cannot exceed the number of poles.

The Hermitian codes over $\mathbb{F}_{q_1^2}$ are the codes defined by the above method using the Hermitian curve as χ:

$$X^{q_1+1} - Y^{q_1} - Y = 0$$

It is well-known that this curve indeed is nonsingular and absolutely irreducible. Furthermore, the curve contains q_1^3 affine $\mathbb{F}_{q_1^2}$-rational points and has genus $\frac{q_1(q_1-1)}{2}$. In this case, the point P_∞ corresponds to the (unique) point at infinity on the homogenization of the Hermitian curve.

In the following it will be assumed that a function field, $\mathbb{F}_q(\chi)$, is given and that $P_1, \ldots, P_n, P_\infty$ are points on a nonsingular and absolutely irreducible curve, χ.

4.3 Prerequisites

For $\ell \geq 2g-1$, $\mathcal{L}(\ell P_\infty)$ is a vector space over $\mathbb{F}_q$ of dimension $\ell - g + 1$. It is well known that $\mathcal{L}(\ell P_\infty)$ has a basis, $\phi_1, \ldots, \phi_{\ell-g+1}$ where the pole order at P_∞ is increasing:

$$\mathbf{v}_{P_\infty}(\phi_1^{-1}) < \mathbf{v}_{P_\infty}(\phi_2^{-1}) < \cdots < \mathbf{v}_{P_\infty}(\phi_{\ell-g+1}^{-1})$$

However, the following theorem (from [13]) shows the existence of bases where the zero multiplicity of a given point – different from P_∞ – is increasing. Furthermore, the proof of the theorem describes a strategy to find these bases.

Theorem 4.1
Let P_i ($i \in \{1, \ldots, n\}$) be a point and let $\ell \geq 2g-1$. Then there exist functions, $\phi_{1,i}, \ldots, \phi_{\ell-g+1,i}$ such that

$$\mathcal{L}(\ell P_\infty) = \text{span}\{\phi_{1,i}, \ldots, \phi_{\ell-g+1,i}\}$$

and furthermore,

$$\mathbf{v}_{P_i}(\phi_{1,i}) < \mathbf{v}_{P_i}(\phi_{2,i}) < \cdots < \mathbf{v}_{P_i}(\phi_{\ell-g+1,i})$$

In the following, such a basis will be called an increasing zero basis with respect to the point P_i. □

Proof:
Suppose that some basis, $B = \{\phi_1, \ldots, \phi_{\ell-g+1}\}$, of $\mathcal{L}(\ell P_\infty)$ is given. Let $B_i := \emptyset$ and do the following:

Let $e := \min\{\mathbf{v}_{P_i}(f) \mid f \in B\}$ and let $A := \{f \in B \mid \mathbf{v}_{P_i}(f) = e\}$.

If $|A| = 1$ then let

$$B := B \backslash A$$

$$B_i := B_i \cup A$$

If $|A| > 1$ then let $a_1, \ldots, a_{|A|}$ be an enumeration of the elements in A and write each a_j as its standard representation:

$$a_j = t_i^e u_j \ , \quad j = 1, \ldots, |A|$$

where $\mathbf{v}_{P_i}(u_j) = 0$ and $\mathbf{v}_{P_i}(t_i) = 1$. Now let

$$B := (B \backslash A) \cup \{t_i^e(u_j(P_i)u_1 - u_1(P_i)u_j) \mid j = 2, \ldots, |A|\}$$

$$B_i := B_i \cup \{a_1\}$$

This ensures that $v_{P_i}(f) > e$ for all $f \in B$.

The process above is repeated $\ell - g + 1$ times until $B = \emptyset$. After this, B_i holds an increasing zero basis of $\mathcal{L}(\ell P_\infty)$ with respect to P_i since B_i is constructed so that two elements cannot have the same valuation in P_i. ■

Recall that the non-negative integers, $\mathbb{N}$, are divided into gaps and non-gaps by calling $s \in \mathbb{N}$ a gap if and only if $\mathcal{L}(sP_\infty) \backslash \mathcal{L}((s-1)P_\infty) = \emptyset$. The number of gaps equals the genus, g, of the curve defining the function field. For $t \in \mathbb{N}$, $\mathbf{g}(t)$ denotes the number of gaps less than or equal to t. That is

$$\mathbf{g}(t) := t - \dim(\mathcal{L}(tP_\infty)) + 1 \tag{4.1}$$

Let R denote the following vector space:

$$R := \bigcup_{i=0}^{\infty} \mathcal{L}(iP_\infty)$$

Suppose that $R = \text{span}\{\phi_i \mid i \geq 1\}$ with the poleorders of the ϕ_i's being strictly increasing. Then $R[z] = \text{span}\{\phi_i z^j \mid i \geq 1 \wedge j \geq 0\}$ (where z is transcendental over $\mathbb{F}_q(\chi)$). We will define a total ordering on these basis functions by associating a non-negative integer – called the weight – to each function. The ordering will be parameterized by the number associated with z. Let this be denoted by $\mathbf{w}(z)$. Then the weight of the basis function $\phi_i z^j$ is given by

$$\mathbf{w}(\phi_i z^j) = \mathbf{v}_{P_\infty}(\phi_i^{-1}) + j\mathbf{w}(z) \tag{4.2}$$

An ordering can now be defined using some lexicographic rule to break ties, for example:

$$\phi_i z^j < \phi_a z^b \Leftrightarrow \rho(\phi_i z^j) < \rho(\phi_a z^b) \vee (\rho(\phi_i z^j) = \rho(\phi_a z^b) \wedge j < b) \tag{4.3}$$

However, in this context only the weighting is important.

ρ is extended to any non-zero function in $R[z]$ by the following definition:

Definition 4.2
Let $f \in R[z]\backslash\{0\}$. Suppose that $f = \sum_{i,j} f_{i,j}\phi_i z^j$ and that $\mathbf{w}(z)$ is given. Then the weight of f is defined as

$$\mathbf{w}(f) = \max\{\rho(\phi_i z^j) \mid f_{i,j} \neq 0\}$$

with $\mathbf{w}$ given by (4.2). □

The following lemma describes the weight of the basis functions:

Lemma 4.3
Suppose that the basis functions of $R[z]$ are enumerated increasingly with respect to the ordering induced by the weighting $\mathbf{w}(z) \geq 2g - 1$:

$$Q_0, Q_1, \ldots$$

with

$$\mathbf{w}(Q_0) \leq \mathbf{w}(Q_1) \leq \ldots$$

Let $j \in \mathbb{N}$ be given. Then let r and t satisfy

$$\binom{r}{2}\mathbf{w}(z) - (r-1)g \leq j < \binom{r+1}{2}\mathbf{w}(z) - rg$$

$$tr - \mathbf{g}(t) \leq j - (\binom{r}{2}\mathbf{w}(z) - (r-1)g) < (t+1)r - \mathbf{g}(t+1)$$

where $\mathbf{g}(t)$ *is given by (4.1).*

The weight of Q_j *is now given by*

$$\mathbf{w}(Q_j) = (r-1)\mathbf{w}(z) + t$$

□

Proof:
Group the basis functions into disjoint sets, $M_1, M_2, \ldots$, where

$$M_i = \{Q_\ell \mid (i-1)\mathbf{w}(z) \leq \mathbf{w}(Q_\ell) < i\mathbf{w}(z)\}$$

Observe that for each t' with $0 \leq t' < \mathbf{w}(z)$ the number of functions in M_i with weight $t'+(i-1)\mathbf{w}(z)$ is exactly i if t' is a non-gap and $i-1$ if t' is a gap. So $|M_i| = i(\mathbf{w}(z)-g)+(i-1)g = i\mathbf{w}(z)-g$ and $|M_1|+|M_2|+\ldots+|M_{i-1}| = \binom{i}{2}\mathbf{w}(z) - (i-1)g$. By the definition of r it is seen that $Q_j \in M_r$ and by the definition of t, $\mathbf{w}(Q_j) = (r-1)\mathbf{w}(z) + t$. ■

Definition 4.4
Let $f \in R[z]$ *with* $f = \sum_{j=0}^{\deg(f)} f_j(z-z_0)^j$ *for some* $z_0 \in \mathbb{F}_q$. *Then the pair,* (P_i, z_0), *is said to be a zero of multiplicity* s *of* f *if*

$$\mathbf{v}_{P_i}(f_j) \geq s - j \text{ for all } j \leq s$$

and $\mathbf{v}_{P_i}(f_j) = s - j$ *for some* $j \leq s$. □

Lemma 4.5
If (P_i, z_0) *is a zero of multiplicity* s *of* f *then*

$$\mathbf{v}_{P_i}(f(t+z_0)) \geq s$$

for any $t \in R$ *with* $\mathbf{v}_{P_i}(t) \geq 1$. □

Proof:
$f(t+z_0) = \sum_{j=0}^{\deg(f)} f_j t^j$ so the Lemma follows from the fact that $\mathbf{v}_{P_i}(f_j t^j) \geq s$ for all j. ■

Lemma 4.6
Let $\phi_{1,i}, \ldots, \phi_{\ell-g+1,i}$ be an increasing zero basis with respect to P_i and consider a non-zero polynomial, $f \in R[z]$. f can then be written as

$$f(z) = \sum_{j=0}^{u} \sum_{k} f_{j,k} \phi_{k,i} z^j$$

with $u \geq \deg(f)$. If

$$f'_{j,k} := \sum_{\ell=j}^{u} f_{\ell,k} \binom{\ell}{j} z_0^{\ell-j} = 0$$

for all $j + k \leq s$ then (P_i, z_0) is a zero of multiplicity at least s of f. □

Proof:
By direct calculation it can be verified that

$$f(z) = \sum_{j=0}^{\deg(f)} f_j (z - z_0)^j$$

where $f_j = \sum_k \phi_{k,i} f'_{j,k}$. Since $\mathbf{v}_{P_i}(\phi_{k,i}) \geq k - 1$ we have that $\mathbf{v}_{P_i}(f_j) > s - j - 1$ which gives the result. ■

4.4 Sudan's algorithm

Let $\mathrm{B}(w, r)$ denote the ball in $\mathbb{F}_q^n$ with center w and radius r:

$$\mathrm{B}(w, r) := \{u \in \mathbb{F}_q^n \mid \mathrm{d}(w, u) \leq r\}$$

The decoding problem for a code, $C \subseteq \mathbb{F}_q^n$, and a received word, $w \in \mathbb{F}_q^n$, can then be specified as calculating the set

$$\mathrm{dec}_\tau(w) := C \cap \mathrm{B}(w, \tau)$$

where $\tau \geq 0$ is an integer. If τ is smaller than half the minimum distance, then $\mathrm{dec}_\tau(w)$ will always contain at most one codeword, however, we will allow τ to be greater than or equal to half the minimum distance. When that is the case, decoding may not be unique since $\mathrm{dec}_\tau(w)$ may contain two or more codewords. This is therefore called list-decoding, and we will

refer to τ as the error-correcting capability of a decoding algorithm which is capable of calculating $\mathrm{dec}_\tau(w)$ for any received word, w.

The version of Sudan's algorithm given below was first presented in [13]. The algorithm can be seen as an extension of the generalization of Sudan's original algorithm to the case of algebraic geometry codes (see [35] and [38]). The extension gives an improved error-correcting capability over the original algorithm at all information rates if the code is sufficiently long. The description used here is from the presentation of Sudan's original algorithm in [8].

Algorithm 4.7
Input: The code $C_{\mathcal{L}}(P_1+\cdots+P_n,(k+g-1)P_\infty)$ *with* $k>g-1$, *a received word,* $w=(w_1,\ldots,w_n)\in\mathbb{F}_q^n$, *and a parameter,* $s\geq 1$.
Output: $\mathrm{dec}_{\tau_s}(w)$.

- *Set* $\mathbf{w}(z):=v_{P_\infty}(\phi_k^{-1})$ *and calculate* r_s *and* t *so that*

$$\tbinom{r_s}{2}\mathbf{w}(z)-(r_s-1)g\leq n\tbinom{s+1}{2}<\tbinom{r_s+1}{2}\mathbf{w}(z)-r_sg$$

$$tr_s-\mathbf{g}(t)\leq n\tbinom{s+1}{2}-(\tbinom{r_s}{2}\mathbf{w}(z)-(r_s-1)g)<(t+1)r_s-\mathbf{g}(t+1)$$

where $\mathbf{g}(t)$ *is given by (4.1). Now*

$$\tau_s=n-\left\lfloor\frac{(r_s-1)\mathbf{w}(z)+t}{s}\right\rfloor-1$$

- *Calculate* $Q(z)\in\mathbb{F}_q(\chi)[z]\backslash\{0\}$ *so that*

  1. *For all* i, Q *has* (P_i,w_i) *as a zero of multiplicity at least* s *(see Definition 4.4).*
  2. $\mathbf{w}(Q)\leq(r_s-1)\mathbf{w}(z)+t$

- *Factorize* Q *into irreducible factors.*

- *If* $z-f$ *divides* Q *and* $f\in\mathcal{L}((k+g-1)P_\infty)$ *and* $\mathrm{d}((f(P_1),\ldots,f(P_n)),w)\leq\tau_s$ *then add* $(f(P_1),\ldots,f(P_n))$ *to the set of candidates. That is*

$$\begin{aligned}\mathrm{dec}_{\tau_s}(w):=\ &\{(f(P_1),\ldots,f(P_n))\mid f\in\mathcal{L}((k+g-1)P_\infty)\wedge\\ &(z-f)|Q\ \wedge\ \mathrm{d}((f(P_1),\ldots,f(P_n)),w)\leq\tau_s\}\end{aligned}$$

□

Any polynomial, Q, satisfying the two conditions in the algorithm will be called a Q-polynomial (with zero multiplicity s) in the following. The correctness of this algorithm must be shown by proving the existence of a Q-polynomial and proving that such a polynomial has the right factors. The existence is given by the following theorem:

Theorem 4.8
A Q-polynomial does exist. □

Proof:
By Lemma 4.6 it is clear that the first condition on a Q-polynomial can be written as $n\binom{s+1}{2}$ homogeneous linear equations with $n\binom{s+1}{2}+1$ unknowns. Lemma 4.3 ensures that a polynomial which satisfies the first condition on a Q-polynomial can be constructed so that the weight does not exceed $(r_s-1)\mathbf{w}(z)+t$ where r_s and t are as in the algorithm. ■

The fact that a Q-polynomial has all the factors corresponding to codewords in $\mathrm{dec}_{\tau_s}(w)$ is proved by the following lemma and theorem:

Lemma 4.9
Let $f \in \mathcal{L}(mP_\infty)$ and suppose that $f(P_i) = w_i$. Then

$$\mathbf{v}_{P_i}(Q(f)) \geq s$$

□

Proof:
Let $p = f - w_i$. Then $p(P_i) = 0$ so $\mathbf{v}_{P_i}(p) \geq 1$. Since (P_i, w_i) is a zero of Q of multiplicity at least s, Lemma 4.5 implies that $\mathbf{v}_{P_i}(Q(p+w_i)) \geq s$. ■

Theorem 4.10
Let Q be a Q-polynomial and suppose that $f \in \mathcal{L}(k+g-1)$ with

$$\mathrm{d}((f(P_1), \ldots, f(P_n)), w) \leq \tau_s.$$

Then $(z-f)|Q$. □

Proof:
Let $h = Q(f)$. Then $\mathbf{w}(h) \leq (r_s-1)\mathbf{w}(z)+t$. By Lemma 4.9 $\mathbf{v}_{P_i}(h) \geq s$

for each value of i where $f(P_i) = w_i$. This happens at least $n - \tau_s$ times. So the total number of zero multiplicities of h is at least

$$\begin{aligned}\sum_{i=1}^{n} \mathbf{v}_{P_i}(h) &\geq s(n - \tau_s) \\ &= s(\left\lfloor \frac{(r_s - 1)\mathbf{w}(z) + t}{s} \right\rfloor + 1) \\ &> (r_s - 1)\mathbf{w}(z) + t \geq \mathbf{v}_{P_\infty}(h^{-1})\end{aligned}$$

This means that h must be the 0-function since the number of zeroes of a non-zero function cannot exceed the number of poles. Therefore, $z - f$ must divide Q. ■

Remark 4.11
Notice that by Lemma 4.3 the degree of Q must be less than r_s. This means that Sudan's algorithm gives at most $r_s - 1$ codewords as output. So

$$|\mathrm{dec}_{\tau_s}(w)| < r_s$$

□

4.5 Calculating increasing zero bases

In principle, the method for calculating increasing zero bases, which is described in the proof of Theorem 4.1 is perfectly fine. However, in practice there are some unsolved problems since it is not trivial to calculate the standard representation of a function and evaluate the unit. In this section both problems are solved in the case of the Hermitian function field.

Suppose that we want to calculate the standard representation of some polynomial, $f \in \mathcal{O}_{P_i}$, with respect to the point $P_i = (x_i, y_i)$ $(i \in \{1, \ldots, n\})$. In this case $x - x_i$ is a valid local parameter in P_i. This is seen from the fact that

$$\begin{aligned}&X^{q_1+1} - Y^{q_1} - Y = \\ &\qquad (X - x_i)^{q_1+1} - (Y - y_i)^{q_1} + x_i(X - x_i)^{q_1} + x_i^{q_1}(X - x_i) - (Y - y_i)\end{aligned}$$

Notice that $y - y_i$ is a local parameter if and only if $x_i \neq 0$.

The assumption that f is a polynomial implies that f is regular in P_i. So the valuation of f in P_i can in principle be calculated by repeatedly dividing f by $x - x_i$ until a unit (a function which evaluates to a non-zero value in P_i) is obtained. So basically we need an explicit method to divide by $x - x_i$ and to determine if a function is a unit.

Notice that a basis for the ring of polynomials, R, in the Hermitian function field is given by

$$\{(x - x_i)^\alpha (y - y_i)^\beta \mid 0 \leq \alpha < q_1 + 1 \wedge 0 \leq \beta\} \quad , \; i \in \{1, \ldots, n\}$$

Dividing a basis element by $x - x_i$ is easy if it contains $x - x_i$ as a factor. If that is not the case we need to calculate $\frac{y-y_i}{x-x_i}$. By using the "shifted" form of the Hermitian curve given above, it is seen that in the Hermitian function field we have

$$\frac{y - y_i}{x - x_i} = \frac{(x - x_i)^{q_1} + x_i(x - x_i)^{q_1 - 1} + x_i^{q_1}}{(y - y_i)^{q_1 - 1} + 1} \tag{4.4}$$

The denominator on the right side is a unit since it evaluates to 1 in P_i. This unit will be used frequently in the following, so let $e := (y - y_i)^{q_1 - 1} + 1$.

The polynomial f whose standard representation is to be determined can now be written as

$$f = \sum_{\alpha=0}^{q_1} \sum_{\beta \in \mathbb{N}} f_{\alpha,\beta} (x - x_i)^\alpha (y - y_i)^\beta \tag{4.5}$$

where $f_{\alpha,\beta} = \sum_{j \in \mathbb{Z}} f_{\alpha,\beta,j} e^j$ with $f_{\alpha,\beta,j} \in \mathbb{F}_{q_1^2}$ and $f_{\alpha,\beta,j} = 0$ for all but a finite number of values of j. It should be mentioned that this representation of f is not unique, however, that will not be a problem in this context. The idea of using this representation is that f can now be divided by $x - x_i$ in such a way that the result is a function which can be written on the same form.

For $(\alpha, \beta) \neq (0, 0)$ it has already been shown how $f_{\alpha,\beta}(x - x_i)^\alpha (y - y_i)^\beta$ can be divided by $x - x_i$. What is missing is a method to calculate $\frac{f_{0,0}}{x-x_i}$ in the case where f is not a unit. Observe that $f(P_i) = \sum_{j \in Z} f_{0,0,j}$ so f is not a unit if and only if $\sum_{j \in Z} f_{0,0,j} = 0$.

Let $s = \min\{j \mid f_{0,0,j} \neq 0\}$. Then

$$f_{0,0} = e^s \sum_{j \in \mathbb{Z}} f_{0,0,j} e^{j-s}$$

where $\sum_{j\in\mathbb{Z}} f_{0,0,j}e^{j-s}$ is a polynomial in y, which is divisible by $y-y_i$, since f is not a unit ($f(P_i)=0$). Let $h := \frac{\sum_{j\in\mathbb{Z}} f_{0,0,j}e^{j-s}}{y-y_i}$. Then

$$\frac{f_{0,0}}{x-x_i} = e^s\frac{y-y_i}{x-x_i}h = e^{s-1}((x-x_i)^{q_1} + x_i(x-x_i)^{q_1-1} + x_i^{q_1})h \qquad (4.6)$$

This leads to the following algorithm for calculating the standard representation of a polynomial in the Hermitian function field:

Algorithm 4.12

Input: Polynomial, f, and point $P_i = (x_i, y_i)$.
Output: u and m so that $m = \mathbf{v}_{P_i}(f)$ and $f = u(x-x_i)^m$.

1. *Initialization:* $f^{(0)} := f = \sum_{\alpha,\beta} f_{\alpha,\beta}(x-x_i)^\alpha(y-y_i)^\beta$, $m := 0$.
2. *If* $\sum_{j\in\mathbb{Z}} f_{0,0,j} \neq 0$ *then stop.*
3. *Use equations (4.6) and (4.4) to calculate*

$$f^{(m+1)} := \frac{f^{(m)}}{x-x_i} =$$

$$\frac{{f_{0,0}}^{(m)}}{x-x_i} + \sum_{\alpha\geq 1,\beta} {f_{\alpha,\beta}}^{(m)}(x-x_i)^{\alpha-1}(y-y_i)^\beta + \sum_{\beta\geq 1} {f_{0,\beta}}^{(m)}(y-y_i)^{\beta-1}\frac{y-y_i}{x-x_i}$$

$$m := m+1$$

4. *Go to step 2.*

□

Notice that this algorithm has only been stated for polynomials since that is what is needed in this context. However, the algorithm can be used to calculate the standard representation of any rational function, $h = \frac{f}{g}$, by using the algorithm on f and g separately. Suppose that the results are $f = f_1 t^m$ and $g = g_1 t^n$. Then $h = \frac{f_1}{g_1}t^{m-n}$.

The above algorithm for calculating standard representations provides all what is needed to implement the method for determining increasing

zero bases in the proof of Theorem 4.1 since the unit of the standard representation is given on a form which can be evaluated directly.

To make the implementation more efficient it may be observed that the increasing zero basis is calculated only for use in the interpolation step. As it will be seen later, this means that in practice a full increasing zero basis is not needed. It will be sufficient with a basis where a given point, P_i, is a zero of increasing multiplicity for the first $s+1$ basis elements (where s is the parameter of Sudan's algorithm). The rest of the basis elements should just have P_i as a zero of multiplicity greater than s. This can be used to speed up the computations. It can also be used to save computer memory since many of the basis elements will just be shifted monomials (on the form $(x-x_i)^\alpha(y-y_i)^\beta$), which can be computed whenever they are needed so they do not have to be stored.

4.6 Interpolation

The goal of the interpolation step is to determine a valid Q-polynomial. As mentioned in Theorem 4.8, such a polynomial must exist in the vector space

$$\text{span}\{Q_0, \ldots, Q_\ell\} \quad , \ \ell := \binom{s+1}{2} n$$

with $Q_0, \ldots, Q_\ell$ being as in Lemma 4.3, and a Q-polynomial can be found by solving a system of linear equations using Gaussian elimination (In the following each of these equations will be referred to as a zero-condition, see Lemma 4.6). However, the system has a special structure, and that can be used to make the calculations more efficient. One method for doing this is described in the following. The method is an application of the Fundamental Iterative Algorithm, along the same lines as the application in [20], Chapter 4. The Fundamental Iterative Algorithm was first presented in [7].

Let $\text{ord} : R[z] \to \mathbb{N} \cup \{-\infty\}$ be given by

$$\begin{aligned}
&\text{ord}(0) = -\infty, \\
&\text{ord}(\text{span}\{Q_0\} \backslash \{0\}) = \{0\} \text{ and} \\
&\text{ord}(\text{span}\{Q_0, \ldots, Q_i\} \backslash \text{span}\{Q_0, \ldots, Q_{i-1}\}) = \{i\}
\end{aligned}$$

for $i > 0$. If $f \in R[z]$ then $\text{ord}(f)$ will be called the order of the function f.

The task of finding a Q-polynomial can now be rephrased as the task of finding a polynomial which satisfies the zero-conditions and which is minimal with respect to ord. It will be safe only to look for such a polynomial in the vector space

$$V = \{fz^j \mid f \in R \wedge 0 \leq j \leq \max\{\deg(Q_j) \mid 0 \leq j \leq \ell\}\}$$

The idea of the algorithm which is presented in the following is to make a partition of V and then consider the zero-conditions one by one while maintaining a list of polynomials – one from each partition class – which all satisfies the zero-conditions considered so far, and which all are minimal within the given partition class.

To do this we need for each point a (small) element of $R[z]$ which has a given pair, (P_i, w_i), as a zero of multiplicity at least s and which whenever it is multiplied by a polynomial gives a result within the same partition class as that polynomial. To make this work, the order of these elements must be the same for all $i \in \{1, \ldots, n\}$. Therefore, choose t as the smallest integer so that

$$\mathrm{ord}(\phi_{j(i),i}) = t \wedge \mathbf{v}_{P_i}(\phi_{j(i),i}) \geq s \quad , \ i = 1, \ldots, n$$

Now construct a partition of V doing the following:

Let

$$A := \mathrm{ord}(V) \backslash \{i \in \mathrm{ord}(V) \mid \exists f \in \mathrm{ord}^{-1}(\{i\}) \exists h \in \mathrm{ord}^{-1}(\{t\}) : h|f\}$$

Furthermore, let $h \in \mathrm{ord}^{-1}(\{t\})$ and define

$$T := \bigcup_{i=0}^{\infty} \mathrm{ord}^{-1}(\mathrm{ord}(h^i))$$

Notice that this definition of T does not depend on the choice of h.

Let A= $\{a_1, \ldots, a_{|A|}\}$ and let

$$G_j := \mathrm{ord}^{-1}(\mathrm{ord}(\{f_j g \mid f_j \in \mathrm{ord}^{-1}(\{a_j\}) \wedge g \in T\}))$$

Now $G_1, \ldots, G_{|A|}$ is a partition of V, and furthermore, if a polynomial in some G_j is multiplied by a polynomial in $\mathrm{ord}^{-1}(\{t\})$ then the result will remain in G_j.

Finally, the following notation is needed: Let $f \in R[z]$ be written as $f = \sum_{j,k} f_{j,k}\phi_{k,i}z^j$ then

$$\mathrm{coef}(f, \phi_{k,i}z^j) := f_{j,k}$$

Now the algorithm can be stated:

Algorithm 4.13
Input: Pairs, $(P_1, w_1), \ldots, (P_n, w_n)$*, and required zero multiplicity,* s*.*
Initialize by setting

$$G := \{g_1, \ldots, g_{|A|}\}$$

so that $\mathrm{ord}(G) = A$ *(which means* $\mathrm{ord}(g_j) = \min \mathrm{ord}(G_j)$ *for* $j = 1, \ldots, |A|$*).*
For $i = 1, \ldots, n$ *do the following:*
For each pair $(j,k) \in \mathbb{N}^2$ *with* $j + k \leq s$ *and* $k \geq 1$ *do the following:*
Let $G^\star := \{g \in G \mid \mathrm{coef}(g(z + w_i), \phi_{k,i}z^j) \neq 0\}$.
If $G^\star \neq \emptyset$ *then choose* $f \in G^\star$ *so that* $\mathrm{ord}(f) = \min \mathrm{ord}(G^\star)$ *and let*

$$G := \{\phi_{j(i),i} f\} \cup \{\mathrm{coef}(f(z+w_i), \phi_{k,i}z^j)g - \mathrm{coef}(g(z+w_i), \phi_{k,i}z^j)f\}$$

After this, the result is given by the polynomial in G *which is smallest with respect to* ord. □

The correctness of this algorithm is proved by induction over the iteration steps by seeing that the set G at any time holds a list of polynomials satisfying the zero-conditions considered so far and minimal with respect to ord within each of the partition classes, $G_1, \ldots, G_{|A|}$. Notice that G is initialized with polynomials which are minimal within each partition class, so the property holds in the beginning. Furthermore, in each step, only the order of the polynomial called f in the algorithm is changed. So the main point is to prove that this polynomial is minimal (satisfying also the new zero-condition) within its partition class after the update. Let $f^{(0)}$ and $f^{(1)}$ denote the polynomial before and after the change respectively.

$f^{(0)}$ is minimal by assumption, furthermore, it can be assumed to be monic. Suppose that g is a monic polynomial satisfying the same zero-conditions as $f^{(1)}$ and that $\mathrm{ord}(g) = \mathrm{ord}(f^{(0)})$. Let $h := g - f^{(0)}$. Then $\mathrm{ord}(h) < \mathrm{ord}(f^{(0)})$, but h does satisfy the previous zero-conditions while not satisfying the new zero-condition, however, since $f^{(0)}$ is the smallest

polynomial with this property, this is a contradiction. So no polynomial with the same order as $f^{(0)}$ can satisfy the zero-conditions considered so far. Therefore it is necessary for $f^{(1)}$ to have greater order than $f^{(0)}$. Since multiplying by a polynomial of order t gives the smallest possible increase in order within the partition class, $f^{(1)}$, is minimal.

When implementing this algorithm it may be noticed that an upper bound on the order of the solution is known. Therefore, an efficiency gain is obtained by discarding polynomials if the order exceeds the upper bound after an update, since such polynomials will never contribute to the solution.

4.7 Factorization

The Q-polynomial is a polynomial in $R[z]$ and therefore, factorization is not so easy. However, fairly simple and efficient methods exist for factoring univariate polynomials over a finite field (see for example [22], Chapter 4). In this section the problem of factoring the Q-polynomial is transformed into a problem of factoring a univariate polynomial over a (large) finite field.

In the case of Hermitian codes, $R = \mathbb{F}_{q_1^2}[X,Y]/\langle X^{q_1+1} - Y^{q_1} - Y\rangle$ and $R = \text{span}\{x^\alpha y^\beta \mid 0 \leq \alpha \leq q_1 \wedge 0 \leq \beta\}$. So seen this way, polynomials in $\mathcal{L}(mP_\infty)$ have degree in x smaller than $q_1 + 1$ and degree in y smaller than some integer, c. Now let $f(Y) \in \mathbb{F}_{q_1^2}[Y]$ with $\deg(f) \geq c$ and

$$D_1 = \mathbb{F}_{q_1^2}[Y]/\langle f(Y)\rangle$$

Furthermore, let $g(X) := X^{q_1+1} - Y^{q_1} - Y \bmod f$ and

$$D_2 = D_1[X]/\langle g(X)\rangle$$

Consider the mappings, $\phi : \mathbb{F}_{q_1^2}[X,Y][z] \to D_1[X][z]$ and $\theta : D_1[X][z] \to D_2[z]$, given by

$$\phi(h_1) = h_1 \bmod f$$

$$\theta(h_2) = h_2 \bmod g$$

It is a well-known fact that these mappings are homomorphisms and that the composition of these mappings, $\theta\phi$, is again a homomorphism. So

suppose that $h \in \mathcal{L}(mP_\infty)$ and that $z - h \mid Q$ then $\theta\phi(z-h) \mid \theta\phi(Q)$ and furthermore, reducing $z - h$ modulo f and g will not change $z - h$ since the degree in Y of f and the degree in X of g is higher than the degrees of h in Y and X respectively. Therefore, in the factorization step, it will be sufficient to factorize $\theta\phi(Q)$, since this will still reveal the factors corresponding to codewords within distance τ_s from the received word.

This will be very useful if $f(Y)$ is chosen so that D_2 is a finite field. That will be the case if and only if D_1 is a finite field (f is irreducible over $\mathbb{F}_{q_1^2}$) and $g(X)$ is irreducible over D_1.

Suppose that f is irreducible so that $D_1 \simeq \mathbb{F}_{q_1^{2c_1}}$, where $c_1 = \deg(f)$. Notice that $g(X) = X^{q_1+1} - (y^{q_1} + y)$ then is a binomial in $\mathbb{F}_{q_1^{2c_1}}[X]$. The question is now if f can be chosen with degree at least c so that g is irreducible. That this is in fact the case is shown in the following.

The following theorem, which is a special case of theorem 3.75 of [22] is needed:

Theorem 4.14
Let $\omega \in \mathbb{F}_{q_1^{2c}} \backslash \{0\}$ and let e denote the order of ω. Then $X^{q_1+1} - \omega$ is irreducible over $\mathbb{F}_{q_1^{2c}}$ if and only if each prime factor of $q_1 + 1$ divides e but not $\frac{q_1^{2c}-1}{e}$. □

Notice that since $q_1 + 1$ divides $q_1^{2c} - 1$ the theorem states that $X^{q_1+1} - \omega$ is irreducible if and only if

$$\gcd(q_1 + 1, \frac{q_1^{2c} - 1}{e}) = 1$$

where e is the order of ω in $\mathbb{F}_{q_1^{2c}} \backslash \{0\}$.

The existence of a polynomial as the one called f above is given by the following theorem:

Theorem 4.15
Let $c_1 \geq 1$ be an integer. Then there exists a polynomial, $f(Y)$, so that $\deg(f) \geq c_1$ and the order, e, of $y^{q_1} + y$ in $\mathbb{F}_{q_1^2}[Y]/\langle f(Y) \rangle$ satisfies

$$\gcd(q_1 + 1, \frac{q_1^{2c_1} - 1}{e}) = 1$$

□

Proof:
This can be done as follows: Let $f_1(Y) \in \mathbb{F}_{q_1^2}[Y]$ be an arbitrary irreducible polynomial of degree c_1 and let $F_q = \mathbb{F}_{q_1^2}[Y]/\langle f_1(Y)\rangle$, where $q = q_1^{2c_1}$. Furthermore, let α be a primitive element of $\mathbb{F}_q$ and set $h(x) = x^{q_1} + x - \alpha \in \mathbb{F}_q[x]$. $h(x)$ must have a root, γ, in some extension field of $\mathbb{F}_q$, and all q_1 roots are given by

$$\gamma + a \ , \quad \text{where } a^{q_1} + a = 0$$

since

$$h(\gamma + a) = (\gamma + a)^{q_1} + (\gamma + a) - \alpha = \gamma^{q_1} + \gamma - \alpha + a^{q_1} + a = 0$$

Notice that for q_1 even:

$$\{a \mid a^{q_1} + a = 0\} = \mathbb{F}_{q_1} \subseteq \mathbb{F}_{q_1^2}$$

and for q_1 odd:

$$\{a \mid a^{q_1} + a = 0\} \subseteq \mathbb{F}_{q_1^2}$$

So in any case the difference of two roots of $h(x)$ is an element of $\mathbb{F}_{q_1^2} \subseteq \mathbb{F}_q$.

Suppose that $\gamma \notin \mathbb{F}_q$. γ^q is also a root of $h(x)$ since $h(\gamma^q) = (h(\gamma))^q = 0$. Therefore, $\gamma^q - \gamma \in \mathbb{F}_{q_1^2}$ and

$$(\gamma^q - \gamma)^q = \gamma^q - \gamma \Rightarrow \gamma^{q^2} - \gamma^q = \gamma^q - \gamma \Rightarrow \gamma^{q^2} = 2\gamma^q - \gamma$$

By induction it is seen that

$$\gamma^{q^i} = i\gamma^q - (i-1)\gamma$$

So for $i = p$: $\gamma^{q^p} = p\gamma - (p-1)\gamma = \gamma$ which means that $\gamma \in \mathbb{F}_{q^p}$. On the other hand, if $\gamma^{q^j} = \gamma$ then $j\gamma^q = j\gamma$. Since $\gamma \notin \mathbb{F}_q$ this implies $p|j$. Therefore, $\mathrm{ord}(\gamma)|q^p - 1$ and $\deg(m_\gamma(x)) = p$ where $m_\gamma(x)$ is the minimal polynomial of γ over $\mathbb{F}_q$.

Let $f(Y)$ be the minimal polynomial of γ over $\mathbb{F}_{q_1^2}$ and suppose that this has degree c. Then $\mathbb{F}_{q_1^2}(\gamma) \simeq \mathbb{F}_{q_1^2}[Y]/\langle f(Y)\rangle = \mathbb{F}_{q_1^{2c}}$ where an isomorphism is given by $\phi(\gamma) = y$. In $\mathbb{F}_{q_1^2}(\gamma)$ the order of $\gamma^{q_1} + \gamma$ is $q_1^{2c_1} - 1$ so in $\mathbb{F}_{q_1^{2c}}$, $y^{q_1} + y$ has order $q_1^{2c_1} - 1$. Now look at

$$\gcd(q_1 + 1, \frac{q_1^{2pc_1} - 1}{q_1^{2c_1} - 1}) = \gcd(p^s + 1, \frac{q^p - 1}{q - 1}) = \gcd(p^s + 1, \sum_{i=0}^{p-1} q^i)$$

Let p_1 be some prime which divides $p^s + 1$. Then $q_1 \equiv -1 \pmod{p_1}$ so $\sum_{i=0}^{p-1} q^i \equiv p \pmod{p_1}$. Since $p_1 \neq p$ this implies that p_1 does not divide $\sum_{i=0}^{p-1} q^i$. Therefore,

$$\gcd(q_1 + 1, \frac{q_1^{2pc_1} - 1}{q_1^{2c_1} - 1}) = 1$$

so $\mathbb{F}_q[X, Y]/\langle X^{q_1+1} - Y^{q_1} - Y, f(Y)\rangle$ is a field. ■

It should be mentioned that experiments indicate that irreducible polynomials with the property described in Theorem 4.15 are rather common, so in practice it seems to be sufficient to generate random irreducible polynomials and check if they have the right property. However, we have no proof that this will always work.

4.8 Example

This section contains an example of decoding a Hermitian code using Sudan's algorithm. The example has been made using an implementation of Sudan's algorithm in C++ based on the ideas presented in the previous sections. The source code of the implementation is available at

`http://www.student.dtu.dk/~p938546/impl.html`

The alphabet of the code used for the example is $\mathbb{F}_4$ with a primitive element, α, satisfying $\alpha^2 + \alpha + 1 = 0$. The Hermitian curve over $\mathbb{F}_4$ is $X^3 - Y^2 - Y = 0$ which has genus 1 and contains the following 8 points:

$$P = \{(0,0), (0,1), (1,\alpha), (1,\alpha^2), (\alpha,\alpha), (\alpha,\alpha^2), (\alpha^2,\alpha), (\alpha^2,\alpha^2)\}$$

The code of the example is $C_{\mathcal{L}}(P_1 + \ldots + P_8, 4P_\infty)$ which is a $(8, 4, 4)$-code, and choosing $s = 6$ we get $\tau_6 = 2$ and $r_6 = 9$ in Sudan's algorithm.

The transmitted codeword in this example is the 0-word. However, suppose that 2 errors occur, so the following word is received:

$$(\alpha^2, 0, 0, \alpha^2, 0, 0, 0, 0)$$

The usual algorithm for decoding Hermitian codes, Sakata's algorithm with majority voting (see [17]), fails to decode this word.

In this case, increasing zero bases are needed for $\mathcal{L}(36P_\infty)$ with respect to each point. For example, the first 8 elements of an increasing zero basis with respect to $P_4 = (1, \alpha^2)$ are as follows ($e := 1 + y$):

$$\begin{aligned}
\phi_{1,4} &:= 1 = (x-1)^0 \\
\phi_{2,4} &:= (x-1) = (x-1)^1 \\
\phi_{3,4} &:= (x-1)^2 = (x-1)^2 \\
\phi_{4,4} &:= (x-1) + (y-\alpha^2) = \\
&\quad (x-1)^3\left((e^{-3}+e^{-2}+e^{-1}) + (e^{-3}+e^{-2})(x-1) + e^{-3}(x-1)^2\right) \\
\phi_{5,4} &:= (x-1)^2 + (x-1)(y-\alpha^2) = \\
&\quad (x-1)^4\left((e^{-3}+e^{-2}+e^{-1}) + (e^{-3}+e^{-2})(x-1) + e^{-3}(x-1)^2\right) \\
\phi_{6,4} &:= (x-1) + (y-\alpha^2) + (x-1)^2(y-\alpha^2) = \\
&\quad (x-1)^5((e^{-4}+e^{-3}+e^{-1}) + e^{-4}(x-1) + e^{-5}(y-\alpha^2)^2 + \\
&\quad e^{-5}(x-1)(y-\alpha^2)^2 + e^{-5}(x-1)^2(y-\alpha^2)^2) \\
\phi_{7,4} &:= (x-1)^2 + (y-\alpha^2)^2 = \\
&\quad (x-1)^6((e^{-6}+e^{-4}+e^{-2}) + e^{-6}(x-1) + e^{-5}(y-\alpha^2) + \\
&\quad (e^{-6}+e^{-4})(x-1)^2 + e^{-5}(x-1)(y-\alpha^2)) \\
\phi_{8,4} &:= (x-1) + (y-\alpha^2) + (x-1)^2 + (y-\alpha^2)^2 + (x-1)(y-\alpha^2)^2 = \\
&\quad (x-1)^7((e^{-5}+e^{-4}+e^{-2}) + e^{-5}(x-1) + e^{-6}(y-\alpha^2)^2 + \\
&\quad e^{-6}(x-1)(y-\alpha^2)^2 + e^{-6}(x-1)^2(y-\alpha^2)^2)
\end{aligned}$$

Furthermore, $\mathbb{F}_{4^3}$ is constructed as $\mathbb{F}_4[Y]/\langle Y^3 + \alpha^2 Y + 1\rangle$ which allows $\mathbb{F}_{4^{3\cdot 3}}$ to be constructed as $\mathbb{F}_{4^3}[X]/\langle X^3 - (y^2+y)\rangle$.

Now the Q-polynomial is calculated using the interpolation method of Section 4.6. The result is the following polynomial in $R[z]$

$$\begin{aligned}
Q = \; & (1 + y + \alpha y^2 + \alpha x^2 y + \alpha^2 x y^2 + \alpha y^3 + \alpha^2 x^2 y^2 + \alpha x y^3 + y^4 + \alpha x^2 y^3 + \\
& \alpha^2 x y^4 + x^2 y^4 + \alpha x y^5 + \alpha^2 x^2 y^5 + \alpha x y^6 + y^7 + \alpha x^2 y^6 + x y^7 + y^8 + \\
& x^2 y^7 + \alpha x y^8 + \alpha y^9 + \alpha^2 x^2 y^8 + x y^9 + \alpha^2 y^1 0 + x^2 y^9) z + (\alpha^2 + \alpha x + \\
& \alpha x^2 + y^2 + \alpha^2 x^2 y + \alpha^2 y^3 + x^2 y^2 + \alpha^2 x y^3 + \alpha^2 y^5 + x^2 y^4 + \alpha^2 y^6 + \\
& \alpha x^2 y^5 + \alpha y^7 + \alpha^2 x^2 y^6 + \alpha x y^7 + y^8 + \alpha^2 x y^8 + \alpha y^9) z^2 + (\alpha^2 + \alpha y + \\
& x y + \alpha^2 x^2 y + x y^2 + \alpha y^3 + x^2 y^2 + \alpha^2 y^4 + \alpha^2 x^2 y^3 + \alpha y^5 + \alpha x^2 y^4 + \\
& \alpha^2 x y^5 + \alpha y^6 + \alpha^2 x^2 y^5 + \alpha^2 x y^6) z^3 + (\alpha + x + \alpha^2 y + x y + \alpha y^2 + \\
& \alpha^2 x^2 y + \alpha x y^2 + y^3 + \alpha x y^3 + \alpha y^4 + x^2 y^3) z^4 + (\alpha + \alpha^2 y + \alpha^2 x y + y^2 + \\
& x^2 y + x y^2 + \alpha^2 x^2 y^2 + \alpha x y^3) z^5 + (1 + \alpha^2 x + \alpha y + \alpha^2 x^2 + \alpha^2 x y + y^2 + \\
& \alpha^2 x^2 y) z^6 + 1 z^7 + (\alpha^2 + \alpha x) z^8
\end{aligned}$$

Which is reduced modulo $Y^3 + \alpha^2 Y + 1$ and $X^3 - (y^2 + y)$ to the following polynomial in $\mathbb{F}_{4^{3\cdot 3}}$:

$$\begin{aligned} Q_{red} = \; & ((\alpha + \alpha y) + (\alpha y + \alpha^2 y^2)x + (\alpha y + y^2)x^2)z + ((\alpha + \alpha^2 y) + (\alpha + \\ & \alpha^2 y)x + (1 + \alpha y^2)x^2)z^2 + ((\alpha^2 y + \alpha^2 y^2) + (\alpha + \alpha y + \alpha^2 y^2)x + \\ & (\alpha^2 + \alpha y + \alpha^2 y^2)x^2)z^3 + ((\alpha^2 + y + \alpha y^2) + (\alpha^2 + \alpha^2 y + \alpha y^2)x + \\ & (\alpha y + y^2)x^2)z^4 + ((\alpha + y) + (\alpha + \alpha y^2)x + (1 + \alpha y)x^2)z^5 + ((y + \\ & \alpha y^2) + (1 + \alpha y + y^2)x + (\alpha + \alpha^2 y^2)x^2)z^6 + ((1 + \alpha^2 y^2) + (\alpha^2 + \\ & \alpha y^2)x + (\alpha + y^2)x^2)z^7 + z^8 \end{aligned}$$

It turns out that this polynomial has three factors of degree one:

$$\begin{aligned} & ((\alpha^2 + \alpha y + y^2) + (\alpha y + \alpha^2 y^2)x + (1 + \alpha^2 y + \alpha y^2)x^2) + z \\ & (\alpha^2 + \alpha^2 x + \alpha^2 x^2) + z \\ & z \end{aligned}$$

The first of these factors does not correspond to a codeword since it is not within $\mathcal{L}(4P_\infty)[z]$. The last two factors corresponds to the codewords

$$\begin{aligned} & (\alpha^2, \alpha^2, \alpha^2, \alpha^2, 0, 0, 0, 0) \\ & (0, 0, 0, 0, 0, 0, 0, 0) \end{aligned}$$

Which both have distance 2 to the received word. So some knowledge of the context is needed in order to choose between the two candidates. However, having a small number, for example two, candidates to choose from is often much more useful than a decoding failure.

4.9 Conclusion

We have demonstrated how to decode Hermitian codes efficiently beyond half the minimum distance using Sudan's algorithm. The main steps are calculation of increasing zero bases, fast interpolation in order to determine the Q-polynomial, and a fast method of factorization. The actual complexity of the overall algorithm remains to be calculated and the extension to more general algebraic geometry codes is a subject for future work.

Chapter 5

Fast bivariate interpolation

R. Refslund Nielsen[1]

Abstract

An efficient method for interpolation of polynomials with coefficients in certain subspaces of a function field in one variable is presented. The method may be applied in the implementation of recent list decoding algorithms. Special cases are presented to give a simple description of the most common applications.

5.1 Introduction

Recently, various results have ben obtained in the area of list decoding algebraic evaluation codes. The first result was a method for list decoding Reed-Solomon codes by Sudan [38]. After this followed a generalization for list decoding one-point algebraic geometry codes by Shokrollahi and Wassermann [35]. Both results were improved by Sudan and Guruswami [13] who developed a method with increased error-correcting capability.

The method was applied by Kötter and Vardy in [21] to attack the problem of soft-decision decoding of BCH, Reed-Solomon, and one-point algebraic geometry codes. List decoding of concatenated codes where the outer code is a one-point algebraic geometry code and the inner code is a Hadamard code was described by Sudan and Guruswami [14].

A modification of the method was used for list decoding one-point algebraic geometry codes for the m-metric in Algorithm 6.9 and Algorithm 6.29. In Chapter 8 decoding in the m-metric was used to develop a

[1]Department of Mathematics, Technical University of Denmark, Bldg. 303, DK-2800, Lyngby, Denmark

decoding method for the class of codes described by Xing and Ling in [42]. Furthermore, in Chapter 7 the method was applied to list decoding of any concatenated code where the outer code is a Reed-Solomon code (the generalization to one-point algebraic geometry codes is straight-forward).

In all cases the underlying decoding algorithm consists of two steps: interpolation and factorization. This paper treats the first step. The task is to find a non-zero bivariate polynomial (a univariate polynomial with coefficients in certain subspaces of a function field of one variable) which satisfies a given set of linear constraints.

In principle, this can be done by solving a system of linear equations, and the expressions developed in Section 5.3.3 shows how to set up the coefficient matrix. With respect to the computational complexity it should be noted that solving the linear equations is *not* necessarily the critical part. In the general case it may be harder to calculate the coefficient matrix. However, in any case the computational complexity of the method described in this paper will be equivalent to that of calculating the coefficient matrix and multiply it with a limited number (the number of elements of the set G in the algorithm) of vectors, which in all cases will be faster than calculating the coefficient matrix and reduce it. The algorithm in this paper can be seen as an application of the Fundamental Iterative Algorithm, [7].

Special cases of the method found in this paper have been presented in Algorithm 3.15 and in Algorithm 4.13. However, this paper introduces a speed-up as well as a clearer proof of the correctness of the method and a description which is general enough to be applied to all the papers mentioned above.

5.2 Algebraic function fields of one variable

This section contains a short summary of definitions and facts about algebraic function fields of one variable which are needed for this paper. For details, refer to [37, Ch. 1] or [15, Sec. 2].

Throughout the paper, $\mathbb{F}_q$, denotes a finite field with q elements and $\mathbb{N} = \{0, 1, 2, \dots\}$ denotes the set of non-negative integers.

5.2.1 Places and divisors

Let $F/\mathbb{F}_q$ be the algebraic function field of one variable defined by a non-singular and absolutely irreducible curve, χ, given by a set of polynomial equations. The elements of $F/\mathbb{F}_q$ may be thought of as rational functions in several variables over $\mathbb{F}_q$ modulo the set of polynomials defining the curve.

Let $\mathbb{P}_F$ denote the set of **places** in F (see [37, Def. I.1.8 and Sec. B.10] for a precise definition). A place may be thought of as a point on the homogenization of χ. If r is the smallest integer such that the point corresponding to a place, P, has all its coordinates in $\mathbb{F}_{q^r}$, then $\deg P := r$ is called the degree of the place P. A place of degree 1 is called an $\mathbb{F}_q$-rational place.

A divisor is a formal sum of places where each place can occur with an integer multiplicity. For example, if $P_1, P_2 \in \mathbb{P}_F$ then $P_1 - 2P_2$ and $3P_1$ are divisors. In general, divisors have the form

$$A = \sum_{P \in \mathbb{P}_F} a_P P$$

where $a_P \in \mathbb{Z}$ and $a_P = 0$ for almost all $P \in \mathbb{P}_F$. Addition is defined on the set of divisors as placewise addition and the set of divisors thereby form a group.

A partial ordering, $\leq$, is defined on the group of divisors as follows:

$$A \leq B \Leftrightarrow \forall P \in \mathbb{P}_F, a_P \leq b_P$$

where A and B are divisors and a_P and b_P are the coefficents of the place P in A and B respectively. Furthermore, the **degree of a divisor**, A, is defined as

$$\deg(A) := \sum_{P \in \mathbb{P}_F} a_P \deg(P).$$

5.2.2 Valuations and evaluation

Let $P \in \mathbb{P}_F$ and let $\mathbf{v}_P$ denote the **valuation** of P. Informally, $\mathbf{v}_P : F \to \mathbb{Z} \cup \{\infty\}$ is a mapping which for a function, $f \in F$, counts with which multiplicity the point corresponding to the place P is a zero of f. So if $\mathbf{v}_P(f) > 0$ then P is said to be a **zero** of f of multiplicity $\mathbf{v}_P(f)$. If $\mathbf{v}_P(f) < 0$ then P is a **pole** of f of multiplicity $-\mathbf{v}_P(f)$.

A valuation, $\mathbf{v}_P$, has the following properties for all $f, g \in F$ ([37, Def. I.1.9]):

1. $\mathbf{v}_P(f) = \infty \Leftrightarrow f = 0$.
2. $\mathbf{v}_P(fg) = \mathbf{v}_P(f) + \mathbf{v}_P(g)$.
3. $\mathbf{v}_P(f+g) \geq \min\{\mathbf{v}_P(f), \mathbf{v}_P(g)\}$ with equality if $\mathbf{v}_P(f) \neq \mathbf{v}_P(g)$.
4. $\exists h \in F, \mathbf{v}_P(h) = 1$.
5. $\mathbf{v}_P(a) = 0, a \in \mathbb{F}_q$.

If a place, P, is not a pole of a function, $f \in F$, then f can be evaluated in the point corresponding to P. The evaluation value is denoted by $f(P)$.

5.2.3 $\mathcal{L}$-spaces, gaps, and bases

The **principal divisor** of a function $f \in F$ is defined as the following formal sum of places:

$$(f) := \sum_{P \in \mathbb{P}_F} \mathbf{v}_P(f) P.$$

For any $f \in F \backslash \{0\}$ we have ([37, The. I.4.11])

$$\deg((f)) = 0.$$

This means that the number of zeroes and poles of any non-zero function in the function field is equal when counted with degree and multiplicity.

Let A be a divisor and define the $\mathcal{L}$-space of A as

$$\mathcal{L}(A) = \{f \in F \mid -A \leq (f)\} \cup \{0\}.$$

Then $\mathcal{L}(A)$ is a vector space over $\mathbb{F}_q$ ([37, Lemma I.4.6]).

In the rest of the text only $\mathcal{L}$-spaces of a multiple of a place of degree 1 will be discussed. For the rest of this section it is straight-forward to generalize the discussion to more general divisors, however, the same is not true for the next section.

Below it will be assumed that $P_\infty \in \mathbb{P}_F$ is a given place of degree 1.

For any $m \in \mathbb{N}$ we have that $\mathcal{L}((m-1)P_\infty) \subseteq \mathcal{L}(mP_\infty)$. The integer m is called a **gap** if equality occurs and a **non-gap** otherwise. Let $\mathbf{g}(m)$ denote the number of gaps smaller than or equal to m. Then

$$\dim(\mathcal{L}(mP_\infty)) = m - \mathbf{g}(m) + 1. \tag{5.1}$$

The total number of gaps is a finite constant, g, which is independent on the choice of P_∞ ([37, Prop. I.4.14 and Prop. I.4.17(b)]). Furthermore, the largest gap is at most $2g-1$ ([37, The. I.5.17]).

Let

$$R := R(P_\infty) = \bigcup_{m=0}^{\infty} \mathcal{L}(mP_\infty) \tag{5.2}$$

denote the vector space of functions with poles only at P_∞.

It is well-known that there exists an **increasing pole basis** of R ([15, Ex. 3.8 and Prop. 3.12]). Namely, a basis, $\phi_0, \phi_1, \ldots$ such that the elements have increasing pole multiplicity in P_∞:

$$0 = \mathbf{v}_{P_\infty}(\phi_0) > \mathbf{v}_{P_\infty}(\phi_1) > \cdots . \tag{5.3}$$

Given a place $P \neq P_\infty$ of degree 1 there also exists an **increasing zero basis** of R with respect to P (for a constructive proof, see Theorem 6.20). Namely a basis, $\phi_{P,0}, \phi_{P,1}, \ldots$ such that the elements have increasing zero multiplicity in P:

$$0 = \mathbf{v}_P(\phi_{P,0}) < \mathbf{v}_P(\phi_{P,1}) < \cdots . \tag{5.4}$$

In this paper it will be assumed that an increasing zero basis is a permutation of an increasing pole basis. Such a basis always exists, however, there may not exist an increasing zero basis which is also an increasing pole basis without reordering the elements.

The Strong Approximation Theorem, [37, Theorem I.6.4], gives the following strong statement about the zero multiplicities of an increasing zero basis:

Lemma 5.1
For any $j \in \mathbb{N}$ we have

$$\mathbf{v}_P(\phi_{P,j}) = j.$$

□

Proof:
By the Strong Approximation Theorem there exists a function, $f \in F$, such that $\mathbf{v}_P(f) = 1$ and $\mathbf{v}_{\hat{P}}(f) \geq 0$ for all $\hat{P} \in \mathbb{P}_F \backslash \{P, P_\infty\}$. Consequently, $f^i \in R$ for all $i \in \mathbb{N}$.

Let $\phi_{P,0}, \phi_{P,1}, \ldots$ be an increasing zero basis of R with respect to the place P. Suppose that there exists some $i \in \mathbb{N}$ such that none of the functions in this basis has P as a zero of multiplicity i. Then f^i is linearly independent of the basis which is a contradiction. This shows that all zero multiplicities are represented in the increasing zero basis and since the zero multiplicities of the basis elements are strictly increasing, the lemma follows. ■

Corollary 5.2
Let $f \in R\backslash\{0\}$ and let $P \neq P_\infty$ be a place of degree 1. Let $\phi_{P,0}, \phi_{P,1}, \ldots$ be an increasing zero basis with respect to P and write f as

$$f = \sum_j f_{P,j} \phi_{P,j}, \quad f_{P,j} \in \mathbb{F}_q.$$

Then $f_{P,j} = 0$ for all $j \leq u$ if and only if $\mathbf{v}_P(f) \geq u$. □

Proof:
Follows by Lemma 5.1 and the properties of a valuation function. ■

5.2.4 Weight

Let $\mathbb{N}_\infty := \mathbb{N} \cup \{-\infty\}$ and extend the usual addition, subtraction, and comparison on $\mathbb{N}$ to $\mathbb{N}_\infty$ such that $a \pm -\infty = -\infty \pm a = -\infty$ and $-\infty \leq a$ for all $a \in \mathbb{N}_\infty$.

Define the map $\rho : R \to \mathbb{N}_\infty$ by

$$\rho(f) = \begin{cases} -\infty & \text{if } f = 0 \\ m & \text{if } f \neq 0 \text{ and } f \in \mathcal{L}(mP_\infty)\backslash\mathcal{L}((m-1)P_\infty). \end{cases}$$

The value $\rho(f)$ — which equals $-\mathbf{v}_{P_\infty}(f)$ — is referred to as the **weight** of f. Notice that the weights of an increasing pole basis, $\rho(\phi_0), \rho(\phi_1), \ldots$ are exactly the non-gaps listed increasingly.

The set of non-gaps, $S = \rho(R\backslash\{0\})$, with the usual addition forms a semigroup. Let $\preceq$ be a partial ordering on S defined as follows for all $s, t \in S$:

$$s \preceq t \Leftrightarrow \exists u \in S, s + u = t.$$

Lemma 5.3
Let $S := \rho(R)$ be the set of non-gaps and let $g < \infty$ denote the number of gaps. If $s \in S$ and $A := \{a \in S \mid s \not\preceq a\}$ then

1. *$|A| = s$ ([15, Lemma 5.15]).*
2. *The following sets form a partition of S:*

$$R_a := \{a + is \mid i \in \mathbb{N}\}, \quad a \in A.$$

□

Proof:

1. Notice that

$$\begin{aligned} a \in A \Leftrightarrow \\ a \in S \wedge \nexists u \in S, s + u = a \Leftrightarrow \\ a \in S \wedge a \notin s + S \Leftrightarrow \\ a \in S \backslash (s + S) \end{aligned}$$

   where $s + S = \{t + s \mid t \in S\}$. So $A = S \backslash (s + S)$.

   Choose c such that $t \in S$ for all $t \geq c$. This is possible since g is finite. Then define the sets

$$\begin{aligned} T &:= \{t \in S \mid t \geq s + c\} \\ U &:= \{u \in S \mid u < s + c\} \\ V &:= \{v \in s + S \mid v < s + c\} \end{aligned}$$

   Then $T \cap U = T \cap V = \emptyset$, $T \cup U = S$, and $T \cup V = s + S$ so

$$|S \backslash (s + S)| = |U| - |V| = s + c - g - (c - g) = s.$$

2. Let $a' \in R_a$. It must be shown that $a' \notin R_b$ for all $b \in A \backslash \{a\}$. For some $i \in \mathbb{N}$ we have $a' = a + is$ by definition of R_a. Suppose that $a' \in R_b$. Then $a' = b + js = a + is$ for some $j \in \mathbb{N}$. If $b < a$ then $j > i$ and $a = b + (j - i)s$ means that $s \preceq a$ which is a contradiction. If $b > a$ then $j < i$ and $b = a + (i - j)s$ means that $s \preceq b$ which is a contradiction. Therefore, if $a' \in R_a$ then $a' \notin R_b$ so the sets are disjoint.

Let $u \in S$. We can always find integers, i and j, such that $u = si + j$ and if i is sufficiently small (for example if $i = 0$) then $j \in S$. Suppose that i is chosen as large as possible such that $j \in S$. This implies that $j - s \notin S$ and, therefore, $s \not\preceq j$ and we have that $j \in A$ and $u \in R_j$. So the union of all the sets equals S.

■

5.3 Polynomials over R

Let y be transcendental over $F/\mathbb{F}_q$. Let $P_\infty \in \mathbb{P}_F$ be a place of degree 1 and let

$$\mathcal{P} := \{P \in \mathbb{P}_F \backslash \{P_\infty\} \mid \deg P = 1\}$$

be the set of all places of degree 1 except P_∞. In the following some properties of the polynomial ring, $R[y]$ — where R is defined as in (5.2) — will be discussed in order to present the interpolation algorithm which is the main result of this paper.

5.3.1 Special bases, weight, and ordering

One basis of $R[y]$ is

$$\mathcal{B} := \{\phi_a y^b \mid a, b \in \mathbb{N}\}. \tag{5.5}$$

Choosing a weight of y we can extend ρ to $R[y]$ as follows: Let $Q = \sum_{b\in\mathbb{N}} Q_b y^b$ with $Q_b \in R$ and $Q_b = 0$ for almost all $b \in \mathbb{N}$. Then define

$$\rho(Q) = \max\{\rho(Q_b) + b\rho(y) \mid Q_b \neq 0\}$$

where $\rho(y)$ is a chosen integer parameter such that $\rho(y) \geq 2$.

The weight induces a total order on $R[y]$ when using the polynomial degree to break ties. For example, as follows:

$$f \leq g \Leftrightarrow \rho(f) < \rho(g) \vee (\rho(f) = \rho(g) \wedge \deg(f) \leq deg(g)) \tag{5.6}$$

for all $f, g \in R[y]$.

Lemma 5.4
Suppose that $\rho(y) \geq 2g$ and let $f_1, f_2, \ldots$ be a list of all the basis elements of (5.5) in increasing order with respect to the order defined in (5.6). Let $j \geq \rho(y)$ be given and let λ and t be the integers satisfying

$$\rho(y)\binom{\lambda}{2} - (\lambda-1)g \leq j-1 < \rho(y)\binom{\lambda+1}{2} - \lambda g \tag{5.7}$$

and

$$\lambda r - \mathbf{g}(r) \leq j - (\rho(y)\binom{\lambda}{2} - (\lambda-1)g) < \lambda(r+1) - \mathbf{g}(r+1) \tag{5.8}$$

where $\mathbf{g}(r)$ is the number of gaps less than or equal to r (see 5.1).

Then

$$\deg(f_j) < \lambda \text{ and } \rho(f_j) = \rho(y)(\lambda-1) + t.$$

□

Proof:
Let $\mathcal{M}(i) := \{\phi_a y^b \mid i\rho(y) \leq \rho(\phi_a y^b) < (i+1)\rho(y)\}$. Then $|\mathcal{M}(0)| = \rho(y) - g$ and $|\mathcal{M}(i)| = |\mathcal{M}(i-1)| + \rho(y)$ for $i \geq 1$. So in general, $|\mathcal{M}(i)| = (i+1)\rho(y) - g$ and

$$\sum_{i=0}^{\lambda-2} |\mathcal{M}(i)| = \sum_{i=0}^{\lambda-2} (i+1)\rho(y) - g = \rho(y)\binom{\lambda}{2} - (\lambda-1)g$$

so $f_j \in \mathcal{M}(\lambda-1)$ where λ satisfies (5.7). This gives that $\deg(f_j) \leq \lambda - 1$.

Listing the elements of $\mathcal{M}(\lambda-1)$ in increasing order, f_j is element number $j - (\rho(y)\binom{\lambda}{2} - (\lambda-1)g)$ on the list. Furthermore, for each integer a with $0 \leq a < \rho(y)$, the number of elements in $\mathcal{M}(\lambda-1)$ of weight $\rho(y)(\lambda-1)+a$ is λ if a is a non-gap and $\lambda - 1$ if a is a gap. So if r is chosen satisfying (5.8) then the weight of f_j is as stated in the lemma. ■

5.3.2 Zero conditions

Let $P \in \mathcal{P}$ and let $f \in R$. Then a basis of $R[y]$ is

$$\{\phi_{P,a}(y-f)^b \mid a, b \in \mathbb{N}\}. \tag{5.9}$$

Any polynomial, $Q \in R[y]$ may be written with respect to the basis in (5.5) as well as the basis in (5.9):

$$Q = \sum_{a,b \in \mathbb{N}} Q_{a,b} \phi_a y^b$$
$$Q = \sum_{a,b \in \mathbb{N}} Q_{a,b}^{(P,f)} \phi_{P,a} (y - f)^b$$

with $Q_{a,b}, Q_{a,b}^{(P,f)} \in \mathbb{F}_q$ for all $a, b \in \mathbb{N}$. Since this represents a change of basis, the coefficients $Q_{a,b}^{(P,f)}$ can be written as a linear combination of the coefficients $Q_{a,b}$.

A **zero condition** on a polynomial $Q \in R[y]$ as above is an equation in the form $Q_{a,b}^{(P,f)} = 0$. In the following, zero conditions will be denoted by $Z(Q) = 0$ where it is understood that $Z(Q) = Q_{a,b}^{(P,f)}$ for some given values of $P \in \mathcal{P}$, $f \in R$, $a \in \mathbb{N}$, and $b \in \mathbb{N}$. So $Z(Q) \in \mathbb{F}_q$. If a zero condition is not related to any single named polynomial then no argument will be specified and the zero condition will be written as $Z = 0$. In that case a zero condition can be seen as a mapping, $Z : R[y] \to \mathbb{F}_q$, defined by $Z(Q) = Q_{a,b}^{(P,f)}$ for any $Q \in R[y]$ and some fixed $(P, f) \in \mathcal{P} \times R$ and $a, b \in \mathbb{N}$.

We will consider lists of zero conditions, $Z_1(Q) = 0, \ldots, Z_m(Q) = 0$, which in all cases will be assumed to satisfy the following: For any $j \in \{1, \ldots, m\}$, if $Z_j(Q) = Q_{a,b}^{(P,f)}$ then $Q_{a',b}^{(P,f)}$ must be among $Z_1(Q), \ldots, Z_{j-1}(Q)$ for all $a' < a$. A list of zero conditions satisfying this will be called **left justified**.

The term "left justified" comes from the following visualization: Suppose that the zero conditions in a list which corresponds to a given place, P, are marked — one by one according to their order in the list — in a usual coordinate system such that a zero condition $Z_j(Q) = Q_{a,b}^{(P,f)}$ is marked by printing an x in the coordinate (a, b). If the list of zero conditions is left justified then each time an x is printed, all the coordinates to the left of the current one must have an x already.

5.3.3 Calculating zero conditions

In general, the relation between the basis in (5.5) and the basis in (5.9) is quite complicated. In this section it is shown how the coefficients of

a polynomial in $R[y]$ with respect to the basis in (5.9) can be calculated given the coefficients with respect to the basis in (5.5). In practice the calculations are done assuming that all polynomials which will be treated lie in a given subspace of $R[y]$ of finite dimension.

Let $P \in \mathcal{P}$. When an increasing zero basis with respect to P is known we can calculate coefficients, $\gamma_{P,a,t} \in \mathbb{F}_q$ such that

$$\phi_a = \sum_t \gamma_{a,P,t}\phi_{P,t}, \quad \text{for } a \in \mathbb{N}$$

and coefficients $\mu_{P,\alpha}^{(t,j)} \in \mathbb{F}_q$ such that

$$\phi_{P,t}\phi_{P,j} = \sum_\alpha \mu_{P,\alpha}^{(t,j)}\phi_{P,\alpha}, \quad \text{for } t, j \in \mathbb{N}.$$

Let $(P, \hat{y}) \in \mathcal{P} \times R$ be given and calculate coefficients $\hat{y}_j^{(b-\beta)} \in \mathbb{F}_q$ such that

$$\hat{y}^{b-\beta} = \sum_j \hat{y}_j^{(b-\beta)}\phi_{P,j}, \quad \text{for } b - \beta \in \mathbb{N}.$$

Then we have

$$\begin{aligned} y^b &= ((y-\hat{y})+\hat{y})^b \\ &= \sum_\beta \binom{b}{\beta}(y-\hat{y})^\beta \hat{y}^{b-\beta} \\ &= \sum_\beta \binom{b}{\beta}(y-\hat{y})^\beta \sum_j \hat{y}_j^{(b-\beta)}\phi_{P,j}. \end{aligned} \tag{5.10}$$

If $Q \in R[y]$ then

$$\begin{aligned} Q &= \sum_{a,b} Q_{a,b}\phi_a y^b \\ &= \sum_{a,b} Q_{a,b} \sum_t \gamma_{a,P,t}\phi_{P,t} \sum_\beta \binom{b}{\beta}(y-\hat{y})^\beta \sum_j \hat{y}_j^{(b-\beta)}\phi_{P,j} \\ &= \sum_{a,b} Q_{a,b} \sum_t \gamma_{a,P,t} \sum_\beta \binom{b}{\beta}(y-\hat{y})^\beta \sum_j \hat{y}_j^{(b-\beta)}\phi_{P,t}\phi_{P,j} \\ &= \sum_{a,b} Q_{a,b} \sum_t \gamma_{a,P,t} \sum_\beta \binom{b}{\beta}(y-\hat{y})^\beta \sum_j \hat{y}_j^{(b-\beta)} \sum_\alpha \mu_{P,\alpha}^{(t,j)}\phi_{P,\alpha} \\ &= \sum_{\alpha,\beta} \phi_{P,\alpha}(y-\hat{y})^\beta \sum_{a,b} Q_{a,b}\binom{b}{\beta} \sum_t \gamma_{a,i,t} \sum_j \hat{y}_j^{(b-\beta)}\mu_{P,\alpha}^{(t,j)} \end{aligned}$$

so for any $\alpha, \beta \in \mathbb{N}$ we have

$$Q_{\alpha,\beta}^{(P,\hat{y})} = \sum_{a,b} Q_{a,b} \binom{b}{\beta} \sum_{t} \gamma_{a,P,t} \sum_{j} \hat{y}_j^{(b-\beta)} \mu_{P,\alpha}^{(t,j)}.$$

Notice that $\binom{b}{\beta} = 0$ for $\beta > b$ and by Lemma 5.1, $\mu_{P,\alpha}^{(t,j)} = 0$ for $\alpha < t + j$. Furthermore, the sum should only be taken over those pairs (a, b) for which $Q_{a,b}$ is non-zero.

in particular, we have the following cases:

1. If $\hat{y} \in \mathbb{F}_q$ then

$$Q_{\alpha,\beta}^{(P,\hat{y})} = \sum_{a,b} Q_{a,b} \binom{b}{\beta} \sum_{t} \gamma_{a,P,t} \hat{y}^{(b-\beta)}.$$

2. If $\phi_{P,t}\phi_{P,j} = \phi_{P,t+j}$ for all $t, j \in \mathbb{N}$ then $\mu_{P,\alpha}^{(t,j)} = 1$ for $\alpha = t + j$ and $\mu_{P,\alpha}^{(t,j)} = 0$ otherwise. So

$$Q_{\alpha,\beta}^{(P,\hat{y})} = \sum_{a,b} Q_{a,b} \binom{b}{\beta} \sum_{j} \gamma_{a,P,\alpha-j} \hat{y}_j^{(b-\beta)}$$

3. If $\hat{y} \in \mathbb{F}_q$ and $\phi_{P,t}\phi_{P,j} = \phi_{P,t+j}$ for all $t, j \in \mathbb{N}$ then

$$Q_{\alpha,\beta}^{(P,\hat{y})} = \sum_{a,b} Q_{a,b} \binom{b}{\beta} \gamma_{a,P,\alpha} \hat{y}^{(b-\beta)}.$$

5.3.4 Interpolation algorithm

The interpolation problem considered in this paper is to find $Q \in R[y]\backslash\{0\}$ such that $\rho(Q)$ is minimal and such that a given set of left justified zero conditions are satisfied. This is done by the following algorithm:

Algorithm 5.5
Input: Left justified zero conditions, $Z_1(Q) = 0, \ldots, Z_m(Q) = 0$.
Output: Minimal polynomial $Q \in R[y]$ satisfying the zero conditions.
Initialization:

1. *Let $G := \{\phi_a y^b \,|\, s \not\preceq a \wedge b < \lambda\}$ where λ is given by (5.7) with $j = m+1$.*

2. *For all* $i \in \{1, \ldots, m\}$, *let* $h_i := \phi_1 - \gamma\phi_0$ *where* $\gamma = \phi_1(P_i)/\phi_0(P_i)$ *with* $Z_i(Q) = Q_{a,b}^{(P_i,y_i)}$ *for some* $P_i \in \mathcal{P}$, $y_i \in F$, *and* $a, b \in \mathbb{N}$.

Iteration:

For $i := 1, \ldots, m$ *do*

Let $f := \min\{g \in G \mid Z_i(g) \neq 0\}$.

Let $G := \{Z_i(f)g - Z_i(g)f \mid g \in G \backslash \{f\}\} \cup \{h_i f\}$

end

Result: min G □

It should be noted that "min" in the algorithm is with respect to the ordering on $R[y]$ defined in (5.6). Furthermore, if all the polynomials in G satisfy the zero conditions $Z_1 = 0, \ldots, Z_i = 0$ at the beginning of the i'th iteration then the i'th iteration is skipped.

The correctness of the algorithm is proved below. The idea is that Lemma 5.4 limits the degree in y which the result may have. Therefore, the result is in one of the partition classes $R_{a,b}$ where $s \not\preceq a \wedge b < \lambda$ with s being the smallest non-gap and λ being given by (5.7) with $j = m + 1$. Throughout the algorithm the set G holds exactly one polynomial from each of these partition classes. The zero conditions are considered one by one during the iteration step. For each zero condition each polynomial in G is updated in such a way that it satisfies all the zero conditions considered so far including the current one and, furthermore, each polynomial stays within its partition class and is minimal within it partition class among the polynomials satisfying the zero conditions considered so far.

It must be shown that the polynomials in the set G always satisfy the zero conditions considered so far. In each iteration this is clear for all polynomials except f by the update expression. The following lemma shows that also the updated polynomial $h_i f$ satisfies the first i zero conditions.

Lemma 5.6

Let $Q \in R[y]\backslash\{0\}$ *be a polynomial which satisfies the left justified zero conditions,* $Z_1(Q) = 0, \ldots, Z_{i-1}(Q) = 0$. *Let* $Z_i(Q) = 0$ *be a new zero condition such that* $Z_1(Q) = 0, \ldots, Z_{i-1}(Q) = 0, Z_i(Q) = 0$ *are left justified. If* s *denotes the smallest positive non-gap then there exists a function,* $h \in R$, *such that* $\rho(h) = s$ *and the zero conditions* $Z_1(hQ) = 0, \ldots, Z_{i-1}(hQ) = 0, Z_i(hQ) = 0$ *are satisfied.* □

Proof:
Suppose that $Z_i(Q) = Q_{a,b}^{(P,\hat{y})}$ and let $h := \phi_1 - \gamma\phi_0$ where $\gamma = \phi_1(P)/\phi_0(P)$ (notice that $\phi_0(P) \neq 0$ since ϕ_0 has pole multiplicity 0 and, therefore, cannot have any zeroes). Then $h(P) = 0$ so $\mathbf{v}_P(h) \geq 1$. Furthermore, $\rho(h) = \rho(\phi_1) = s$.

Let $j \in \{1, \ldots, i\}$ and suppose that $Z_j(Q) = Q_{a',b'}^{(P',\hat{y}')}$. Notice that

$$h \cdot Q = \sum_{\alpha,\beta \in \mathbb{N}} Q_{\alpha,\beta}^{(P',\hat{y}')}(h\phi_{P',\alpha})(y - \hat{f}')^{\beta}$$

In all cases, $h\phi_{P',\alpha} \in R$, and $\mathbf{v}_{P'}(h\phi_{P',\alpha}) \geq \mathbf{v}_{P'}(\phi_{P',\alpha})$ so the zero conditions $Z_1(hQ) = 0, \ldots, Z_{i-1}(hQ) = 0$ are satisfied by Corollary 5.2.

If $P' = P$ then $\mathbf{v}_{P'}(h\phi_{P',\alpha}) > \mathbf{v}_{P'}(\phi_{P',\alpha})$ so by Corollary 5.2 also $Z_i(hQ) = 0$ is satisfied since the zero conditions are left justified. ■

The following theorem shows that each polynomial in G is minimal after each iteration. Since the partition classes of the polynomials partition the space in which the result is known to exist and since the overall smallest polynomial is picked as the result, this theorem shows that the algorithm gives the correct output. The following notation is needed in the proof of the theorem:

Let $f \in R[y]\backslash\{0\}$ and write f as

$$f = \sum_{a,b} f_{a,b}\phi_a y^b.$$

Let α, β be chosen such that $\phi_\alpha y^\beta = \max\{\phi_a y^b \mid f_{a,b} \neq 0\}$. Then define

$$LC(f) := f_{\alpha,\beta}$$
$$LM(f) := \phi_\alpha y^\beta.$$

Theorem 5.7
After the i'th iteration of the algorithm, the set G is a set of polynomials such that each polynomial is minimal within it partition class among the polynomials which satisfies the zero conditions $Z_1 = 0, \ldots, Z_i = 0$. □

Proof:
After the initialization, each of the polynomials in G is minimal within its partition classes.

Let $i \in \{1, \ldots, m\}$. Let $G^{(0)}$ and $f^{(0)}$ denote the values of G and f before the i'th iteration (where f is the polynomial chosen in the i'th iteration) and let $G^{(1)}$ and $f^{(1)}$ denote the values after the i'th iteration.

Assume that $G^{(0)}$ is a set of polynomials such that each polynomial is minimal within it partition class among the polynomials which satisfies the zero conditions $Z_1 = 0, \ldots, Z_{i-1} = 0$. The polynomials are then updated in the iteration and only the weight of f changes, so by assumption, any $g \in G^{(1)} \backslash \{f^{(1)}\}$ is minimal within its partition class among the polynomials satisfying the zero conditions $Z_1 = 0, \ldots, Z_i = 0$.

The updated value, $f^{(1)}$, introduces the smallest possible increase in weight within the partition class of $f^{(0)}$. Therefore, it is sufficient to show that this increase is necessary. That is, that no polynomial, $\hat{g}$, exists in the partition class of $f^{(0)}$ which is smaller than $f^{(1)}$ and satisfies the zero conditions $Z_1(\hat{g}) = 0, \ldots, Z_i(\hat{g}) = 0$. By assumption, the weight of $\rho(\hat{g}) = \rho(f^{(0)})$. Therefore,

$$\hat{g}_0 := LC(\hat{g}) f^{(0)} - LC(f^{(0)}) \hat{g}$$

is a non-zero polynomial satisfying $Z_1(\hat{g}_0) = 0, \ldots, Z_{i-1}(\hat{g}_0) = 0$, but not $Z_i(\hat{g}_0) = 0$. However, g_0 is smaller than $f^{(0)}$ so by assumption it must belong to another partition class and there must be some polynomial, $g_0 \in G^{(0)}$, such that $LM(g_0) = LM(\hat{g}_0)$ which satisfies the zero conditions $Z_1(g_0) = 0, \ldots, Z_i(g_0) = 0$. Then

$$\hat{g}_1 := LC(\hat{g}_0) g_0 - LC(g_0) \hat{g}_0$$

is a polynomial smaller than g_0 and satisfying $Z_1(\hat{g}_1) = 0, \ldots, Z_{i-1}(\hat{g}_1) = 0$, but not $Z_i(\hat{g}_1) = 0$. So, there must be some polynomial, $g_1 \in G^{(0)}$, such that $LM(g_1) = LM(\hat{g}_1)$ and which satisfies the i zero conditions. Thus

$$\hat{g}_2 := LC(\hat{g}_1) g_1 - LC(g_1) \hat{g}_1$$

is a polynomial smaller than g_1 and satisfying $Z_1(\hat{g}_2) = 0, \ldots, Z_{i-1}(\hat{g}_2) = 0$, but not $Z_i(\hat{g}_2) = 0$.

Continuing like this we can get a strictly decreasing infinite sequence, $\hat{g}_0, \hat{g}_1, \hat{g}_2, \ldots$ of polynomials. However, this is a contradiction and, therefore, no polynomial like $\hat{g}$ can exist.

The theorem now follows by induction. ■

Notice that it follows from the proof above that if we choose a set of partition classes, $\{R_{a_1,b_1}, \ldots, R_{a_u,b_u}\}$ and initialize G as a set containing a minimal polynomial from each of the partition classes then the result of the algorithm will be a minimal polynomial among the polynomials in the union of the partition classes which satisfy the given zero conditions.

For example, if G is initialized as follows:

$$G := \{\phi_a y^b \mid s \not\preceq a \wedge b < 2\}$$

then a minimal polynomial of degree at most 1 is found which satisfy the given zero conditions. This can be used to modify a list decoding algorithm based on Algorithm 5.5 into an algorithm for unique decoding.

5.4 Interpolation with zero multiplicities

Let size : $R \to \mathbb{N}$ be defined as follows: set size$(0) := 1$ and for any $f \in R\backslash\{0\}$, set size$(f) := u$ where

$$f = \sum_{j=0}^{u-1} f_j \phi_j, \quad f_j \in \mathbb{F}_q$$

with $f_{u-1} \neq 0$.

Definition 5.8
Let $(P, \hat{y}) \in \mathcal{P} \times R$ and let $Q \in R[y]\backslash\{0\}$ where y is transcendental over $F/\mathbb{F}_q$. Let Q be written as

$$Q = \sum_{a,b} Q_{a,b}^{(P,\hat{y})} \phi_{P,a} (y - \hat{y})^b, \quad Q_{a,b}^{(P,\hat{y})} \in \mathbb{F}_q$$

where $\phi_{P,0}, \phi_{P,1}, \ldots$ is an increasing zero basis of R with respect to the place P.

Let $r, s \in \mathbb{N}$ be given integers. If

$$Q_{a,b}^{(P,\hat{y})} = 0 \text{ for all } a, b \in \mathbb{N} \text{ with } a/r + b < s$$

and

$$Q_{a,b}^{(P,\hat{y})} \neq 0 \text{ for some } a, b \in \mathbb{N} \text{ with } a/r + b = s$$

*then the pair $(P, \hat{y})$ is said to be a r-**zero of multiplicity** s of Q. If $r = 1$ then $(P, \hat{y})$ is a **zero of multiplicity** s of Q.* □

Notice that for any pair $(P, \hat{y}) \in \mathcal{P} \times R$ and integers, $s, r \in \mathbb{N}$ with $r \geq \text{size}(\hat{y})$ the requirement that a polynomial $Q \in R[y] \backslash \{0\}$ has the pair as a r-zero of multiplicity at least s can be written as $r\binom{s+1}{2}$ left justified zero conditions on Q.

Therefore, we have the following algorithm. Notice that the weight $\rho(y)$ is an additional parameter of the algorithm.

Algorithm 5.9
Input: Pairs $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{P} \times R$ *and corresponding parameters and multiplicities* $(r_1, s_1), \ldots, (r_n, s_n) \in \mathbb{N}^2$. *The pairs must satisfy that* $\rho(y) \geq r_i \geq \text{size}(y_i)$ *for* $i = 1, \ldots, n$.
Output: Minimal polynomial $Q \in R[y]$ *such that each pair* (x_i, y_i) *is a* r_i*-zero of* Q *of multiplicity at least* r_i *for* $i = 1, \ldots, n$.
Initialization:

1. *Let* $G := \{\phi_a y^b \mid s \not\leq a \wedge b < \lambda\}$ *where* λ *is given by (5.7) with* $j = 1 + \sum_{i=1}^{n} r_i\binom{s_i+1}{2}$.
2. *For all* $i \in \{1, \ldots, n\}$, *let* $h_i := \phi_1 - \gamma\phi_0$ *where* $\gamma = \phi_1(P_i)/\phi_0(P_i)$.

Iteration:
For $i := 1, \ldots, n$ *do*
For $\beta = 0, \ldots, s_i - 1$ *do*
For $\alpha = 0, \ldots, r_i(s_i - \beta - 1)$ *do*
Let $f := \min\{g \in G \mid g_{\alpha,\beta}^{(P_i,y_i)} \neq 0\}$.
Let $G := \{f_{\alpha,\beta}^{(P_i,f_i)} g - g_{\alpha,\beta}^{(P_i,y_i)} f \mid g \in G \backslash \{f\}\} \cup \{h_i f\}$
end
end
end
Result: min G □

The correctness of the algorithm follows from Theorem 5.7.

5.5 Special cases

In this section some special cases are presented where the special structure of the function field can be used to write some of the calculations involved in the general algorithm in a more explicit form.

5.5.1 The rational function field

In this case we have $R = \mathbb{F}_q[x]$ and an increasing pole basis is given by $\phi_j = x^j$ for $j \in \mathbb{N}$. An increasing zero basis with respect to any $x_i \in \mathbb{F}_q$ is given by $\phi_{i,j} = (x - x_i)^j$ for $j \in \mathbb{N}$.

Let $Q \in \mathbb{F}_q[x, y]$ with

$$Q = \sum_{a,b} Q_{a,b} x^a y^b, \quad Q_{a,b} \in \mathbb{F}_q$$

and let $(x_i, y_i) \in \mathbb{F}_q^2$. The situation is that of Case 3 in Sec. 5.3.3. Furthermore, by Eqn. 5.10 we have

$$\phi_a = x^a = \sum_{\alpha} \binom{a}{\alpha} x_i^{a-\alpha} (x - x_i)^\alpha$$

so $\gamma_{a,i,\alpha} = \binom{a}{\alpha} x_i^{a-\alpha}$ and

$$Q_{\alpha,\beta}^{(x_i,y_i)} = \sum_{a,b} Q_{a,b} \binom{b}{\beta} \binom{a}{\alpha} x_i^{a-\alpha} x_i^{(b-\beta)}.$$

All in all we have the following interpolation algorithm.

Algorithm 5.10
Input: Distinct pairs $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{F}_q^2$ *and corresponding multiplicities* $s_1, \ldots, s_n \in \mathbb{N}$.
Output: Minimal polynomial $Q \in \mathbb{F}_q[x, y]$ *having the given pairs as zeroes of at least the given multiplicities.*
Initialization:
Let $G := \{y^b \mid b < \lambda\}$ *where* λ *is given by (5.7) with* $j = 1 + \sum_{i=1}^n \binom{s_i+1}{2}$.
Iteration:
For $i := 1, \ldots, n$ *do*
For $\beta = 0, \ldots, s_i - 1$ *do*
For $\alpha = 0, \ldots, s_i - \beta - 1$ *do*
Let $f := \min\{g \in G \mid g_{\alpha,\beta}^{(x_i,y_i)} \neq 0\}$.
Let $G := \{f_{\alpha,\beta}^{(x_i,y_i)} g - g_{\alpha,\beta}^{(x_i,y_i)} f \mid g \in G \backslash \{f\}\} \cup \{(x - x_i) f\}$
end
end
end
Result: min G □

Now suppose that $x_i \in \mathbb{F}_q$ and $y_i \in \mathbb{F}_q[x]$ with $\deg(y_i) < r$ where r is some given positive integer. This is the situation of Case 2 in Sec. 5.3.3 so

$$Q_{\alpha,\beta}^{(P_i,y_i)} = \sum_{a,b} Q_{a,b} \binom{b}{\beta} \sum_j \binom{a}{\alpha - j} x_i^{a-(\alpha-j)} y_{i,j}^{(b-\beta)} \tag{5.11}$$

where $y_{i,j}^{(b-\beta)} \in \mathbb{F}_q$ such that

$$y_i^{b-\beta} = \sum_j y_{i,j}^{(b-\beta)} (x - x_i)^j.$$

We then have the following algorithm.

Algorithm 5.11
Input: Distinct pairs $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{F}_q \times \mathbb{F}_q[x]$ *and corresponding parameters and multiplicities* $(r_1, s_1), \ldots, (r_n, s_n) \in \mathbb{N}^2$ *such that* $\rho(y) \geq r_i > \deg y_i$ *for* $i = 1, \ldots, n$.
Output: Minimal polynomial $Q \in \mathbb{F}_q[x, y]$ *having each pair* (x_i, y_i) *as a* r_i*-zero of multiplicity at least* s_i *for* $i = 1, \ldots, n$.
Initialization:
Let $G := \{y^b \mid b < \lambda\}$ *where* λ *is given by (5.7) with* $j = 1 + \sum_{i=1}^n r_i \binom{s_i+1}{2}$.
Iteration:
For $i := 1, \ldots, n$ *do*
For $\beta = 0, \ldots, s_i - 1$ *do*
For $\alpha = 0, \ldots, r_i(s_i - \beta) - 1$ *do*
Let $f := \min\{g \in G \mid g_{\alpha,\beta}^{(x_i,y_i)} \neq 0\}$.
Let $G := \{f_{\alpha,\beta}^{(x_i,y_i)} g - g_{\alpha,\beta}^{(x_i,y_i)} f \mid g \in G \backslash \{f\}\} \cup \{(x - x_i) f\}$
end
end
end
Result: min G □

5.5.2 The Hermitian function field

The Hermitian function field is defined by the Hermitian curve over a finite field of size $q = w^2$:

$$\chi : X^{w+1} - Z^w - Z = 0$$

Let x and z denote the cosets in the function field containing X and Z respectively. Let $(x_i, z_i) \in \mathbb{F}_q^2$ be a point on the curve and let P_i be the corresponding place of the function field. We then have that

$$\begin{aligned} &x^{w+1} - y^w - y = \\ &\quad (x - x_i)^{w+1} + x_i(x - x_i)^w + x_i^w(x - x_i) - (z - z_i)^w - (z - z_i) = 0 \end{aligned} \tag{5.12}$$

From this it is seen ([15, Sec. 2.2]) that $x - x_i$ is a local parameter (that is $\mathbf{v}_{P_i}(x - x_i) = 1$).

The special place P_∞ corresponds to the unique point at infinity on the homogenization of χ. Defining $\rho(x) := w$ and $\rho(z) := w + 1$ determines a weight function, ρ, on R ([15, Ex. 3.8 and Prop. 3.17]). An increasing pole basis consists of the following elements ordered with increasing weight:

$$\{x^a z^b \mid 0 \leq a \leq w \wedge 0 \leq b\}.$$

Furthermore, an increasing zero basis can be found explicitly:

Theorem 5.12
Let $\phi_{i,a+(w+1)b} := (x - x_i)^a((z - z_i) - x_i^r(x - x_i))^b$ for $0 \leq a \leq w$ and $0 \leq b$. Then $\mathcal{B} := \{\phi_{i,0}, \phi_{i,1}, \dots\}$ is an increasing zero basis of R with respect to the place P_i. □

The following lemma is needed to prove the theorem:

Lemma 5.13
Let P_i be the place corresponding to the point (x_i, z_i) and let $\phi = (z - z_i) - x_i^w(x - x_i)$. Then

$$\mathbf{v}_{P_i}(\phi) = w + 1.$$

□

Proof:
By Eqn. 5.12,

$$\begin{aligned} \phi &= (x - x_i)^{w+1} + x_i(x - x_i)^w - (z - z_i)^w \\ &= (x - x_i)^{w+1} + x_i(x - x_i)^w - (x - x_i)^{w(w+1)} - x_i^w(x - x_i)^{w^2} - \\ &\qquad\qquad x_i^{w^2}(x - x_i)^w + (z - z_i)^{w^2} \\ &= (x - x_i)^{w+1} - (x - x_i)^{w(w+1)} - x_i^w(x - x_i)^{w^2} + (z - z_i)^{w^2}. \end{aligned}$$

■

Proof of the theorem:
Notice that $\phi_{i,a+(w+1)b} = x^a z^b + f(x,z)$ where all terms in $f(x,z)$ have weight smaller than $\rho(x^a z^b)$ so $\rho(\phi_{i,a+(w+1)b}) = \rho(x^a z^b)$. This shows that $\mathcal{B}$ is a permutation of an increasing pole basis.

Furthermore, Lemma 5.13 gives that $\mathbf{v}_{P_i}(\phi_{i,a+(w+1)b}) = a + (w+1)b$ which means that the zero multiplicity of the elements of $\mathcal{B}$ is increasing. ■

All in all we have the following algorithm:

Algorithm 5.14
Input: Distinct pairs $(P_1, y_1), \ldots, (P_n, y_n) \in \mathcal{P} \times R$ *and corresponding parameters and multiplicities* $(r_1, s_1), \ldots, (r_n, s_n) \in \mathbb{N}^2$ *such that* $\rho(y) \geq r_i \geq \text{size}(y_i)$ *for* $i = 1, \ldots, n$.
Output: Minimal polynomial $Q \in R[y]$ *having each pair* (P_i, y_i) *as a* r_i*-zero of multiplicity at least* s_i *for* $i = 1, \ldots, n$.
Initialization:
Let $G := \{z^a y^b \mid a < w \wedge b < \lambda\}$ *where* λ *is given by (5.7) with* $j = 1 + \sum_{i=1}^{n} r_i \binom{s_i+1}{2}$.
Iteration:
For $i := 1, \ldots, n$ *do*
For $\beta = 0, \ldots, s_i - 1$ *do*
For $\alpha = 0, \ldots, r_i(s_i - \beta) - 1$ *do*
Let $f := \min\{g \in G \mid g_{\alpha,\beta}^{(P_i,y_i)} \neq 0\}$.
Let $G := \{f_{\alpha,\beta}^{(P_i,y_i)} g - g_{\alpha,\beta}^{(P_i,y_i)} f \mid g \in G \backslash \{f\}\} \cup \{(x - x_i)f\}$
end
end
end
Result: min G □

where each place, P_i, in the input corresponds to the point $(x_i, z_i) \in \mathbb{F}_q^2$.

Chapter 6

A Class of Sudan-decodable Codes

R. Refslund Nielsen[1]

Abstract

In this paper Sudan's algorithm is modified into an efficient method to list-decode a class of codes which can be seen as a generalization of Reed-Solomon codes. The algorithm is specialized into a very efficient method for unique decoding. The code construction can be generalized based on algebraic-geometry codes and the decoding algorithms are generalized accordingly. Comparisons with Reed-Solomon and Hermitian codes are made.

Index terms: Sudan's Algorithm, decoding, Reed-Solomon codes, algebraic-geometry codes.

6.1 Introduction

Reed-Solomon codes are often used in practice due to the fact that they can be decoded efficiently and have the optimal minimum distance for the lengths and dimensions where a Reed-Solomon code exists. During the last decades much effort has been put into the construction of codes with lengths and dimensions not obtainable for Reed-Solomon codes while maintaining a good minimum distance. The study of algebraic geometry codes has lead to very promising results.

However, the minimum distance is not the only measure of the usability of a code. For practical purposes it is important that there exist an efficient decoding method to make use of the error-correcting capability, and it

[1]The author is with the Department of Mathematics, Technical University of Denmark, Building 303, DK-2800 Lyngby, Denmark. E-mail: R.R.Nielsen@mat.dtu.dk

is important that error-patterns which are likely to occur in the actual application are usually corrected by the decoder.

For example consider a (n, k) Reed-Solomon code over $\mathbb{F}_{2^m}$. $\mathbb{F}_{2^m}$ can be seen as a vector space of dimension m over $\mathbb{F}_2$, so the code can be seen as a (mn, mk) code over $\mathbb{F}_2$. The minimum distance of the Reed-Solomon code is optimal over $\mathbb{F}_{2^m}$, but the minimum distance of the binary code could be considerably less than for other codes. This means that the Reed-Solomon codes might not correct as many random binary errors as for example a BCH-code. A reason why Reed-Solomon codes are still widely used even though the underlying communication channel is binary is that errors are often likely to happen in bursts, so a bit has higher probability of being erroneous if the previous bit was erroneous too, and a code over a larger alphabet handles this situation better than a binary code.

In [32] a series of new distance functions on vectors over finite sets is introduced and some codes which are good with respect to this distance are constructed. However, decoding methods are not discussed. This paper provides efficient methods for unique decoding and for list-decoding of the codes presented in [32] which are based on Reed-Solomon and algebraic-geometry codes.

The paper is organized as follows: Section 2 describes the construction based on Reed-Solomon codes and Section 3 introduces the so-called r-distance. In Section 4 a list decoding algorithm based on Sudan's algorithm is presented and specialized into a simple algorithm for unique decoding. In Section 5 comparisons to Reed-Solomon codes are discussed and in Section 6 it is shown how the codes can be encoded systematically. Section 7 defines some notation on algebraic function fields and generalizes the code construction using this notation. In Section 8 the decoding algorithms are generalized and Section 9 is the conclusion.

6.2 Construction

Let $\mathbb{F}_q$ denote a finite field with q elements and suppose that

$$P := \{P_1, \ldots, P_{n'}\} \subseteq \mathbb{F}_q \text{ with } |P| = n' \tag{6.1}$$

Consider a polynomial, $f \in \mathbb{F}_q[x]$, with $f = \sum_{j=0}^{\deg(f)} f_j x^j$. Given some $P_i \in P$ we can write

$$f = \sum_{j=0}^{\deg(f)} f_{j,i}(x - P_i)^j$$

and it is seen by direct calculation that

$$f_{j,i} = \sum_{j'=j}^{\deg(f)} f_{j'} P_i^{j'-j} \binom{j'}{j} \tag{6.2}$$

It is useful to observe that

$$(x - P_i)^j | f \Leftrightarrow \forall j' < j (f_{j',i} = 0) \tag{6.3}$$

Definition 6.1
Let r be a positive integer and let $0 < k \leq rn'$. Then define the following error-correcting code:

$$C(P, r, k) = \{f(P, r) \mid \deg(f) < k\}$$

with P being as in (6.1) and

$$f(P, r) := (f_{0,1}, \ldots, f_{r-1,1}; f_{0,2}, \ldots, f_{r-1,2}; \ldots; f_{0,n'}, \ldots, f_{r-1,n'})$$

□

Notice that for $r = 1$ a Reed-Solomon code is obtained.

Furthermore, it is useful to notice that if $f(P, r) = (c_0, \ldots, c_{n-1})$ then for any i and j with $1 \leq i \leq n'$ and $0 \leq j < r$ we have

$$f_{j,i} = c_{(i-1)r+j}$$

Theorem 6.2 ([32], theorem 6)
$C(P, r, k)$ is a $\mathbb{F}_q$-linear code with length $n := rn'$ and dimension k. □

Proof:
The block length is n by construction and that the code is linear follows from the fact that $(f + \alpha g)_{j,i} = f_{j,i} + \alpha g_{j,i}$ for $f, g \in \mathbb{F}_q[x]$ and $\alpha \in \mathbb{F}_q$. To prove that the dimension is k consider a polynomial $f \in \mathbb{F}_q[x] \backslash \{0\}$

with $\deg(f) < k$. Suppose that $f(P, r)$ is the zero vector. This implies that $\prod_{i=1}^{n'}(x - P_i)^r$ divides f, but this is a polynomial of degree rn', which contradicts the assumption that f is non-zero and of degree less than $k \leq rn'$. ■

Notice that the polynomial 1 gives a word of weight n' so the minimum distance is at most n' for all k. So for $k < (r-1)n'$, $C(P, r, k)$ is not a good error-correcting code in the traditional sense, however, as it will be seen later that does not prevent it from performing well in certain situations.

Example 6.3
Let ω be a primitive element of $\mathbb{F}_4$ with $\omega^2 + \omega + 1 = 0$ and let $P := \{0, 1, \omega, \omega^2\}$. Then $C(P, 2, 4)$ is a $(8, 4)$ code over $\mathbb{F}_4$. Suppose that

$$\begin{aligned} f &= 1 + \omega x + \omega x^2 + x^3 \\ &= \omega^2(x-1) + \omega^2(x-1)^2 + (x-1)^3 \\ &= \omega + (x-\omega) + (x-\omega)^3 \\ &= \omega + (x-\omega^2)^2 + (x-\omega^2)^3 \end{aligned}$$

then

$$f(P, 2) = (1, \omega; 0, \omega^2; \omega, 1; \omega, 0)$$

□

6.3 r-distance

As mentioned in the previous section the minimum distance of the code $C(P, r, k)$ is normally bad with respect to the usual Hamming distance. In this section a distance which will be called r-distance is introduced and the properties of $C(P, r, k)$ with respect to r-distance are analyzed. The r-distance was first mentioned in [32].

In $C(P, r, k)$ codewords consists of n' chunks of r field elements where each chunk corresponds to an element in P. This structure is reflected in the following definition of r-distance:

Definition 6.4
Let r be a positive integer and let $u, v \in \mathbb{F}_q^n$ with $n = rn'$ for some integer n'.

For $i \in \{1, \ldots, n'\}$ *define the r-similarity,* $\mathrm{s}_r(u, v, i)$*, and the r-distance,* $\mathrm{d}_r(u, v, i)$*, between* u *and* v *with respect to the* i*'th chunk as follows:*

$$\begin{aligned}\mathrm{s}_r(u, v, i) &:= \max\{j \in \{0, \ldots, r\} \mid u_{(i-1)r+j'} = v_{(i-1)r+j'} \\ &\qquad\qquad \textit{for all } j' \textit{ with } 0 \leq j' < j\} \\ \mathrm{d}_r(u, v, i) &:= r - \mathrm{s}_r(u, v, i)\end{aligned}$$

Furthermore, define the r-similarity, $\mathrm{s}_r(u, v)$*, and the r-distance,* $\mathrm{d}_r(u, v)$*, between* u *and* v*:*

$$\begin{aligned}\mathrm{s}_r(u, v) &:= \textstyle\sum_{i=1}^{n'} \mathrm{s}_r(u, v, i) \\ \mathrm{d}_r(u, v) &:= \textstyle\sum_{i=1}^{n'} \mathrm{d}_r(u, v, i)\end{aligned}$$

□

Let $f \in \mathbb{F}_q[x]$ and $u \in F_q^n$. The following short notations will then be used:

$$\begin{array}{ll}\mathrm{s}_r(f, u, i) := \mathrm{s}_r(f(P, r), u, i) & \mathrm{s}_r(f, u) := \mathrm{s}_r(f(P, r), u) \\ \mathrm{d}_r(f, u, i) := \mathrm{d}_r(f(P, r), u, i) & \mathrm{d}_r(f, u) := \mathrm{d}_r(f(P, r), u)\end{array} \tag{6.4}$$

For example, for all $f \in F_q[x]$ we have $\mathrm{d}_r(f, f(P, r)) = 0$ and

$$\mathrm{d}_r(f, f(P, r) + (0, 1, 0, \ldots, 0)) = r - 1$$

if $r > 1$. Furthermore, $\mathrm{d}_r(f, w) = n - \mathrm{s}_r(f, w)$. For $r = 1$ the usual Hamming distance is obtained so $\mathrm{d}_1 = \mathrm{d}$.

Theorem 6.5
d_r *is a distance function on* $\mathbb{F}_q^n$*.* □

Proof:
Let $u, v, w \in \mathbb{F}_q^n$. $\mathrm{d}_r(u, v) \geq 0$ and it is straight-forward to see that $\mathrm{d}_r(u, v) = 0 \Leftrightarrow u = v$ and that $\mathrm{d}_r(u, v) = \mathrm{d}_r(v, u)$. Furthermore, notice that for $i \in \{1, \ldots, n'\}$ if $\mathrm{s}_r(u, w, i) = r$ then $\mathrm{d}_r(u, w, i) = 0$ so in this case it is trivial that $\mathrm{d}_r(u, w, i) \leq \mathrm{d}_r(u, v, i) + \mathrm{d}_r(v, w, i)$. If $j := \mathrm{s}_r(u, w, i) < r$ then $u_{(i-1)r+j} \neq w_{(i-1)r+j}$ so $v_{(i-1)r+j} \neq u_{(i-1)r+j}$ or $v_{(i-1)r+j} \neq w_{(i-1)r+j}$ and therefore $\mathrm{s}_r(u, v, i) \leq j$ or $\mathrm{s}_r(v, w, i) \leq j$. This implies that $\mathrm{d}_r(u, v, i) \geq \mathrm{d}_r(u, w, i)$ or $\mathrm{d}_r(v, w, i) \geq \mathrm{d}_r(u, w, i)$ for all i so $\mathrm{d}_r(u, w) \leq \mathrm{d}_r(u, v) + \mathrm{d}_r(v, w)$. ■

The following theorem (a special case of [32], theorem 6]) gives the minimum r-distance of $C(P, r, k)$:

Theorem 6.6
If $u, v \in C(P, r, k)$ *with* $u \neq v$ *then* $\mathrm{d}_r(u, v) \geq n - k + 1$ *and* $\mathrm{d}_r(w, 0) = n - k + 1$ *for some* $w \in C(P, r, k)$. □

Proof:
Let $f, g \in \mathbb{F}_q[x]$ be polynomials with degrees less than k so that $u = f(P, r)$ and $v = g(P, r)$. Notice that for each $i \in \{1, \ldots, n'\}$ and $j = 0, \ldots, r-1$

$$(f - g)_{j,i} = f_{j,i} - g_{j,i} = u_{(i-1)r+j} - v_{(i-1)r+j}$$

so $(f - g)_{j,i} = 0$ if $j < \mathrm{s}_r(u, v, i)$ and therefore (6.3) gives:

$$(x - P_i)^{\mathrm{S}_r(u,v,i)} | (f - g) \tag{6.5}$$

which means that a polynomial of degree $\mathrm{s}_r(u, v)$ divides $f - g$. Suppose that $\mathrm{d}_r(u, v) \leq n - k$. Then $\mathrm{s}_r(u, v) \geq k$ implying that $f = g$ and consequently $u = v$ which is false by assumption so $\mathrm{d}_r(u, v) \geq n - k + 1$.

Let $h \in \mathbb{F}_q[x] \backslash \{0\}$ be an arbitrary non-zero polynomial of degree at most $k-1$. For any $i \in \{1, \ldots, n'\}$ and $j = 0, \ldots, r-1$ the equation $h_{j,i} = 0$ is a homogeneous linear equation in the k coefficients of h by equation 6.2. So if $j_1, \ldots, j_{n'} \in \{0, \ldots, r\}$ with $\sum_{i=1}^{n'} j_i = k - 1$ then h can be constructed so that $h_{j'_i,i} = 0$ for all $i \in \{1, \ldots, n'\}$ and $j'_i < j_i$. By definition 6.4, $\mathrm{d}_r(h, 0) \leq n - k + 1$ and by the above $\mathrm{d}_r(h, 0) = n - k + 1$. ■

It can be shown (see [32]) that the minimum r-distance in the above theorem is the greatest possible given the code length and number of codewords.

6.4 Decoding

In [13] V. Guruswami and M. Sudan presented an algorithm to decode Reed-Solomon codes beyond half the minimum distance by allowing the output to be a (small) list of codewords closest to the received word. In this section the method in [13] will be generalized to a list decoding method for $C(P, r, k)$. However, first some notation is needed.

Let $M := \{x^\alpha y^\beta \in \mathbb{F}_q[x, y] \mid (\alpha, \beta) \in \mathbb{N}^2\}$ be the set of monomials in $\mathbb{F}_q[x, y]$. A **monomial ordering** is a binary relation, $<_m$, on M, which satisfies the following:

- $<_m$ is a total ordering on M.

- $\forall f, g, h \in M(f <_m g \Rightarrow fh <_m gh)$
- $<_m$ is a well-ordering.

One monomial ordering is the **lexicographic order**. The lexicographic order with $y < x$ is defined by

$$x^\alpha y^\beta <_l x^{\alpha'} y^{\beta'} \Leftrightarrow \alpha < \alpha' \vee (\alpha = \alpha' \wedge \beta < \beta')$$

The lexicographic order with $x < y$ is defined by exchanging x and y in the above definition.

Let $f(x, y) \in \mathbb{F}_q[x, y] \backslash \{0\}$ with $f = \sum_{\alpha,\beta} f_\alpha^{(\beta)} x^\alpha y^\beta$. Then the (a, b)-**weighted degree** of $f(x, y)$ is given by

$$\deg^{(a,b)}(f) := \max\{\alpha a + \beta b \mid f_\alpha^{(\beta)} \neq 0\}$$

where $a \in \mathbb{N}$ is called the weight of x and $b \in \mathbb{N}$ is called the weight of y. For any choice of a and b we may define $\deg^{(a,b)}(0) := -\infty$. In general, $\deg^{(a,b)}(f)$ is called the weight of f.

Given a weighted degree, $\deg^{(a,b)}$, and a lexicographic order, $<_l$, a corresponding **weighted degree lexicographic order** can be defined on M by

$$\begin{aligned} f <_w g \Leftrightarrow &\deg^{(a,b)}(f) < \deg^{(a,b)}(g) \vee \\ &(\deg^{(a,b)}(f) = \deg^{(a,b)}(g) \wedge f <_l g) \end{aligned}$$

for all $f, g \in M$.

The following lemma describes the weight of the monomials:

Lemma 6.7
Consider the polynomial ring $\mathbb{F}_q[x, y]$ with the weighted degree $\deg^{(1,k-1)}$ for some integer $k > 1$ and let the monomials be ordered by a corresponding $<_w$ order. Suppose that

$$m_0 <_w m_1 <_w m_2 <_w \ldots$$

is an increasing list of all the monomials in $F_q[x, y]$. Then

$$\deg^{(1,k-1)}(m_j) = \left\lfloor \frac{j}{b} + \frac{(b-1)(k-1)}{2} \right\rfloor \tag{6.6}$$

where b satisfies

$$\binom{b}{2} \leq \frac{j}{k-1} < \binom{b+1}{2}$$

□

Proof:
Group the monomials into the disjoint sets, $M_1, M_2, \ldots$, where

$$M_c = \{m_{j'} \mid (c-1)(k-1) \leq \deg^{(1,k-1)}(m_{j'}) < c(k-1)\}$$

Then $|M_c| = c(k-1)$ so $|M_1| + |M_2| + \ldots + |M_{c-1}| = \binom{c}{2}(k-1)$. Since $\binom{b}{2}(k-1) \leq j < \binom{b+1}{2}(k-1)$, $m_j \in M_b$. The smallest monomial in M_b has weighted degree $(b-1)(k-1)$ and for each a with $(b-1)(k-1) \leq a < b(k-1)$ there are exactly b monomials with weighted degree a in M_b. If the monomials of M_b are listed increasingly with respect to $<_w$ then m_j is monomial number $j - \binom{b}{2}(k-1)$ so the weighted degree of m_j must be

$$\begin{aligned}\deg^{(1,k-1)}(m_j) &= (b-1)(k-1) + \left\lfloor \frac{j-\binom{b}{2}(k-1)}{b} \right\rfloor \\ &= \left\lfloor \frac{j}{b} + \frac{(b-1)(k-1)}{2} \right\rfloor\end{aligned}$$

■

The following definition turns out to be useful:

Definition 6.8
Let $w = (w_0, \ldots, w_{n-1}) \in \mathbb{F}_q^n$ with $n = rn'$. Then define

$$w^{(i)}(x) := \sum_{j=0}^{r-1} w_{(i-1)r+j}(x - P_i)^j$$

□

Notice that for any $f \in \mathbb{F}_q[x]$,

$$(x - P_i)^{\mathrm{S}_r(f,w,i)} | (f - w^{(i)}(x)) \tag{6.7}$$

Furthermore, if $Q(x,y) = \sum_{\alpha=0}^{\deg_y(Q)} Q^{(\alpha)} y^\alpha$ then Q can be written as follows:

$$Q(x,y) = \sum_{\alpha=0}^{\deg_y(Q)} Q^{(\alpha,i)}(y - w^{(i)}(x))^\alpha \ , \quad Q^{(\alpha,i)} \in \mathbb{F}_q[x]$$

Algorithm 6.9
As input take the code, $C(P,r,k)$, a received word, w, and a parameter, $s \geq 1$.

Let

$$\ell_s := \left\lfloor \frac{n\binom{s+1}{2}}{b_s} + \frac{(b_s-1)(k-1)}{2} \right\rfloor$$

where b_s satisfies

$$\binom{b_s}{2} \leq \frac{n\binom{s+1}{2}}{k-1} < \binom{b_s+1}{2}$$

Determine $Q(x,y) \in \mathbb{F}_q[x,y]\backslash\{0\}$ so that

$$\deg^{(1,k-1)}(Q) \leq \ell_s$$

and furthermore for $i \in \{1,\ldots,n'\}$, $\alpha \in \{0,\ldots,s-1\}$, and $j \in \{0,\ldots,r(s-\alpha)-1\}$,

$$Q_{j,i}^{(\alpha,i)} = 0 \tag{6.8}$$

Next, let

$$\tau_s := n - \left\lfloor \frac{\ell_s}{s} \right\rfloor - 1$$

and find all factors of Q of the form $y-f$ with $\deg(f) < k$. If $\mathrm{d}_r(f,w) \leq \tau_s$ then include f in the output list. □

The claim is now that the output list contains all codewords, $f(P,r)$, in $C(P,r,k)$ where $\mathrm{d}_r(f,w) \leq \tau_s$. To prove that the algorithm works, it must be proven that the polynomial Q exists and that it has the right factors.

Theorem 6.10
$Q(x,y)$ satisfying the conditions above exists. □

Proof:
Equation 6.8 gives $n'(rs^2 - r(0+1+\ldots+s-1)) = n\binom{s+1}{2}$ conditions on the polynomial Q. Each of these conditions is a homogeneous linear equation in the coefficients of Q. By Lemma 6.7 there are at least $n\binom{s+1}{2}+1$ monomials in $\mathbb{F}_q[x,y]$ with weighted degree at most ℓ_s, so there are $n\binom{s+1}{2}+1$ unknown coefficients. It is a well-known fact from linear algebra that such a system of equations has a non-zero solution. ■

Lemma 6.11
If $f \in \mathbb{F}_q[x]$ with $\deg(f) < k$ then $(x - P_i)^{s\mathrm{S}_r(f,w,i)}$ divides $Q(x, f)$. □

Proof:
Let

$$R(x) := Q(x, f) = \sum_{\alpha=0}^{b_s-1} (f - w^{(i)}(x))^{\alpha} Q^{(\alpha,i)}(x)$$

By (6.7) we have that

$$(x - P_i)^{\alpha \mathrm{S}_r(f,w,i)} | (f - w^{(i)}(x))^{\alpha}$$

For $\alpha \in \{0, \ldots, s-1\}$ (6.8) ensures that

$$(x - P_i)^{r(s-\alpha)} | Q^{(\alpha,i)}(x)$$

Therefore,

$$(x - P_i)^{r(s-\alpha)+\alpha \mathrm{S}_r(f,w,i)} | (f - w^{(i)}(x))^{\alpha} Q^{(\alpha,i)}(x)$$

This proves the lemma since $r(s-\alpha) + \alpha \mathrm{s}_r(f, w, i) \geq \mathrm{s}_r(f, w, i)(s-\alpha) + \alpha \mathrm{s}_r(f, w, i) = s\mathrm{s}_r(f, w, i)$. ■

Theorem 6.12
If a codeword $f(P, r) \in C(P, r, k)$ has $\mathrm{d}_r(f, w) \leq \tau_s$ then $(y - f) | Q$. □

Proof:
By Lemma 6.11 a polynomial of degree $s\mathrm{s}_r(f, w)$ divides $Q(x, f)$, but

$$\mathrm{d}_r(f, w) \leq n - \left\lfloor \frac{\ell_s}{s} \right\rfloor - 1 \Rightarrow \mathrm{s}_r(f, w) \geq \left\lfloor \frac{\ell_s}{s} \right\rfloor + 1$$

and $\deg(Q(x, f)) \leq \ell_s < s\mathrm{s}_r(f, w)$. So $Q(x, f) = 0$ and $y - f$ is therefore a factor of Q. ■

The theorem below gives an idea of the size of τ_s in Algorithm 6.9 and corresponds to the result of [13]. The proof, which is omitted here, is similar to the proof of Theorem 3.31 of [26].

Theorem 6.13
If n and s are sufficiently large then

$$\frac{\tau_s}{n} \approx 1 - \sqrt{\frac{k}{n}}$$

□

The following theorem gives an upper bound on the size of the output list:

Theorem 6.14
The number of codewords returned by Algorithm 6.9 is less than b_s. □

Proof:
By the proof of Lemma 6.7 the maximal degree in y of Q is $b_s - 1$. Therefore, Q can have at most $b_s - 1$ factors of the form $y - f$ so the number of codewords returned by the algorithm is at most $b_s - 1$. ■

The list decoding algorithm can easily be modified into an efficient algorithm for unique decoding of the code $C(P, r, k)$ up to half the minimum r-distance. This algorithm, which can be seen as a generalization of the Welch-Berlekamp algorithm for decoding of Reed-Solomon codes (see for example [25] or [40]), is described in the following.

To modify Algorithm 6.9 into an algorithm for unique decoding, the parameter s is set to 1, and instead of calculating b_s as described in the algorithm, b_s is set to the constant value 2. Furthermore, the Q-polynomial is not allowed to hold terms of degree greater than 1 in y. This gives the following algorithm:

Algorithm 6.15
As input take the code $C(P, r, k)$ *and the received word,* $w \in \mathbb{F}_q^n$.

Let $Q(x, y) = Q^{(0)}(x) + yQ^{(1)}(x) \in \mathbb{F}_q[x, y] \backslash \{0\}$ *satisfy that*

$$\deg(Q^{(0)}) \leq \left\lceil \frac{n+k}{2} \right\rceil - 1 \text{ and } \deg(Q^{(1)}) \leq \left\lfloor \frac{n-k}{2} \right\rfloor$$

and furthermore for each $1 \leq i \leq n'$ *and* $j < r$*:*

$$Q_{j,i}^{(0)} + \sum_{j'=0}^{j} w_{(i-1)r+j'} Q_{j-j',i}^{(1)} = 0 \tag{6.9}$$

If there exists a codeword $f(P, r) \in C(P, r, k)$ *with* $\mathrm{d}_r(f, w) \leq \left\lfloor \frac{n-k}{2} \right\rfloor$ *then* $f = \frac{-Q^{(0)}}{Q^{(1)}}$. □

The proof that this method indeed works as promised will be omitted here since it is very similar to the proof of Algorithm 6.9. However, an example of using the above algorithm is given below.

Example 6.16
This continues Example 6.3. Suppose that the codeword

$$f(P,2) = (1,\omega;0,\omega^2;\omega,1;\omega,0)$$

is sent but $w = (1,\omega^2;0,\omega^2;\omega,\omega^2;\omega,0)$ *is received. Then* $\mathrm{d}_2(f,w) = 2$ *and since* $\lfloor \frac{8-4}{2} \rfloor = 2$ *Algorithm 6.15 should be able to reconstruct* f *from* w.

Solving the system of linear equations in (6.9) gives the following polynomial, Q*:*

$$Q(x,y) = \omega x + \omega x^2 + x^3 + x^5 + (\omega x + x^2)y$$

and

$$\frac{\omega x + \omega x^2 + x^3 + x^5}{\omega x + x^2} = x^3 + \omega x + \omega x + 1 = f$$

So Algorithm 6.15 indeed corrects the 2 errors successfully. □

6.5 Comparing with Reed-Solomon codes

$\mathbb{F}_{q^r}$ can be seen as a r-dimensional vector space over $\mathbb{F}_q$. So suppose that $k = rk'$ for some integer k' and consider each chunk of r elements in a codeword of $C(P,r,k)$ as an element in $\mathbb{F}_{q^r}$. The following theorem gives the main parameters of this code:

Theorem 6.17
$C(P,r,rk')$ is a code over $\mathbb{F}_{q^r}$ of length n', having $q^{rk'}$ codewords, and minimum distance $n' - k' + 1$. □

Proof:
The code length is given by the construction. The number of codewords is equal to the number of codewords in the code seen as a $\mathbb{F}_q$-code namely $q^{rk'}$, however, it should be noticed that the code is not necessarily $\mathbb{F}_{q^r}$-linear. To find the minimum distance consider two polynomials $f, g \in \mathbb{F}_q[x]$, both with degree less than k. Let $(F_1,\ldots,F_{n'}) = f(P,r)$ with $F_i \in \mathbb{F}_{q^r}$ for all i and similarly $(G_1,\ldots,G_{n'}) = g(P,r)$. Suppose that $F_i = G_i$. Then $f_{0,i} = g_{0,i},\ldots,f_{r-1,i} = g_{r-1,i}$ so $(x-P_i)^r | f-g$. This means that if $F_i = G_i$ for k' values of i then a polynomial of degree $k = rk'$ divides $f-g$, but

this implies that $f = g$. Therefore two different codewords can be equal on at most $k' - 1$ positions, which shows that the minimum distance is at least $n' - k' + 1$. Equality holds by the Singleton bound. ■

The theorem shows that $C(P, r, rk')$ has the same main parameters as a (n', k') Reed-Solomon code over $\mathbb{F}_{q^r}$, but what about the error correcting capability of the two codes when using Algorithm 6.15 for unique decoding of $C(P, r, rk')$ and some decoder to decode the Reed-Solomon code up to half the minimum distance? - In the Reed-Solomon code errors will be $\mathbb{F}_{q^r}$-errors, each one corresponding to r $\mathbb{F}_q$-errors. However, in the code $C(P, r, rk')$, error-correcting starts from the point in the affected $\mathbb{F}_{q^r}$ symbol where the error actually starts. The effect of this is that some "fractional" $\mathbb{F}_{q^r}$-errors can be recognized, namely errors which only affect the last part of a $\mathbb{F}_{q_r}$ symbol. For example, on average, random bit-errors will only count as half an $\mathbb{F}_{q^r}$ error where a full error must be corrected by the Reed-Solomon code. Burst errors of length slightly greater than one $\mathbb{F}_{q^r}$-symbol will on average only count as around $\frac{3}{2}$ errors compared to 2 errors in the Reed-Solomon code. However, it should be noted that usually $n' \ll q^r$, so the Reed-Solomon code considered here is very short.

Now compare using a (n', k') Reed-Solomon code over $\mathbb{F}_q$ with using $C(P, r, rk')$, and suppose that rk' information symbols are to be transmitted. r RS-codewords or one $C(P, r, rk')$-codeword will be needed. Let $t := \lfloor \frac{n'-k'}{2} \rfloor$ and suppose that rt errors occur. Which code has the highest probability of correcting the errors? There are $\binom{rn'}{rt}$ error patterns in total, but none of the codes will correct all of them if $r > 1$.

The RS-code will require the errors to be located so that exactly t errors occur in each chunk of n' elements. The number of error patterns with that property is $\binom{n'}{t}^r$ (in each chunk t errors can occur in $\binom{n'}{t}$ different patterns).

The $C(P, r, rk')$-code will require the errors to happen so that only the last part of each chunk of r elements is affected. To see in how many ways this can happen, suppose that rt errors are added one by one, each time one of the n' chunks is selected to receive the error and the error will have to be located at the last correct position in the chunk. The number of ways to do this is at most $\binom{n'+rt-1}{rt}$ and equality holds only if $t \leq 1$ because if a chunk receives r errors, it will not be able to contain more errors. Experiments indicate that $\binom{n'+rt-1}{rt}$ is normally smaller than $\binom{n'}{t}^r$ and when that is the case using r RS-codewords gives a higher probability of decoding rt

randomly positioned errors than using a $C(P, r, rk')$-codeword.

However, it should be emphasized that the error-correcting profile is significantly different for the two codes. Consider an example where $n' = 4$, $k' = 2$, and $r = 2$ (and $q \geq 4$). So 4 $\mathbb{F}_q$ symbols of information can either be sent as 2 codewords of a $(4, 2)$ RS-code or as one codeword of $C(P, 2, 4)$. Let the two RS-codewords be denoted by $a = (a_0, \ldots, a_3)$ and $b = (b_0, \ldots, b_3)$, and the $C(P, 2, 4)$-codeword be denoted by $c = (c_0, \ldots, c_7)$. In the codeword c it is possible to select 4 elements (c_1, c_3, c_5, and c_7) with the property that any pattern of 2 errors happening among these 4 symbols can be corrected. It is not possible to select 4 elements of a and b which has this property, because at least 2 of the elements will always be from the same codeword. On the other hand, if a and b are interleaved so that $(a_0, b_0, a_1, b_1, a_2, b_2, a_3, b_3)$ is sent then 2 consecutive errors will always be corrected. It is not possible to arrange the elements of the word c in such a way that the same is achieved, because if an error happens in c_j with $j \in \{0, 2, 4, 6\}$ then a second error is only guaranteed to be corrected if it occurs in c_{j+1}. Therefore, if an error happens at one of these symbols, then a second error in the previous or in the following symbol will give an error-pattern which is not guaranteed to be corrected.

In the above all error-patterns were assumed to occur with a probability only depending on the weight of the error-pattern. However, this may not always be the case. Consider the following example.

Example 6.18
Let $\mathbb{F}_q = \{\omega_0, \ldots, \omega_{q-1}\}$ and suppose that a vector in $\mathbb{F}_q^{rn'}$ is transmitted as n' integers where a chunk $(\omega_{j_0}, \ldots, \omega_{j_{r-1}})$ is transmitted as $c := \sum_{\ell=0}^{r-1} j_\ell q^{r-1-\ell}$. What is received is $c + e$ where $|e|$ is a small integer. In this case errors are most likely to affect the right-most element, and in general if an element is erroneous then all the elements to the right of that element in the same chunk are almost always erroneous too. This means that the r-distance and the Hamming distance between a codeword and a received word are usually the same. So in this case using a $C(P, r, k)$-code corresponds to using a MDS-code of length rn' which, however, does not exist if $n' = q$ and $r > 1$. □

Finally, a word about complexity. Suppose that we have an implementation of Sudan's algorithm which runs in time $O(n^2)$ where n is the code

length. Then decoding a $C(P, r, k)$-codeword will be $O(r^2n^2)$ while decoding r RS-codewords will be $O(rn^2)$. So decoding of the $C(P, r, k)$ code is generally slower than for the similar RS code.

6.6 Systematic encoding

When using an error-correcting code in practice, it is often desired that encoding can be done systematically. That is, if the code has dimension k then k fixed positions in a codeword contains the information word and the rest of the positions contains check values. In this section a method to encode systematically for the code $C(P, r, k)$ is described. The main part is the following lemma. Notice that the proof is constructive.

Lemma 6.19
Let $j_1, \ldots, j_{n'} \in \{0, \ldots, r-1\}$ be chosen so that $\sum_{i=1}^{n'} j_i = k$. For each $i \in 1, \ldots, n'$ and $j \in 0, \ldots, j_i - 1$ there exists a polynomial $F^{(i,j)} \in \mathbb{F}_q[x]$ with $\deg(F^{(i,j)}) < k$ so that for all $i' \in 1, \ldots, n'$ and $j' \in 0, \ldots, j_{i'} - 1$,

$$F^{(i,j)}_{i',j'} = \begin{cases} 0 & \text{if } i \neq i' \vee j \neq j' \\ 1 & \text{if } i = i' \wedge j = j' \end{cases}$$

□

Proof:
Define

$$B^{(i)} := \frac{\prod_{i'=1}^{n'} (x - P_{i'})^{j_{i'}}}{(x - P_i)^{j_i}}$$

and for $j \in \{0, \ldots, j_i - 1\}$ let

$$B^{(j,i)} := \frac{(x - P_i)^j B^{(i)}}{B^{(i)}(P_i)}$$

Now $\deg(B^{(j,i)}) < k$ and for $i' \neq i$ and $j' < j_{i'}$, $B^{(j,i)}_{j',i'} = 0$. Furthermore, $B^{(j,i)}_{j',i} = 0$ for $j' < j$ and $B^{(j,i)}_{j,i} = 1$. For $j = j_i - 1, \ldots, 0$ define inductively

$$F^{(j,i)} := B^{(j,i)} - \sum_{j'=j+1}^{j_i - 1} B^{(j,i)}_{j',i} F^{(j',i)}$$

Then $F^{(j,i)}$ has the properties stated above by construction. ■

Letting $j_1, \ldots, j_{n'}$ and $F^{(i,j)}$ be as in the lemma above, encoding can now be done systematically. Let $m \in \mathbb{F}_q^k$ denote the information word (the message) with

$$m = (m_{0,1}, \ldots, m_{j_1,1}; \ldots; m_{0,n'}, \ldots, m_{j_{n'},n'})$$

If

$$f = \sum_{i=1}^{n'} \sum_{j=0}^{j_i-1} m_{j,i} F_{j,i}$$

then $\deg(f) < k$ and $f(P, r)$ holds the information word on the k positions determined by the j_i's.

6.7 Construction based on AG-codes

Let χ be a nonsingular absolutely irreducible curve over $\mathbb{F}_q$ and let

$$P_1, \ldots, P_{n'}, P_\infty$$

be $\mathbb{F}_q$-rational points on χ. The curve defines an algebraic function field, $\mathbb{F}_q(\chi)$, with a discrete valuation, $\mathbf{v}_{P_i} : \mathbb{F}_q(\chi) \to \mathbb{Z} \cup \{\infty\}$, corresponding to each point $(i = 1, \ldots, n', \infty)$.

Recall that a function, $f \in \mathbb{F}_q(\chi)$ is called regular in the point P_i if $\mathbf{v}_{P_i}(f) \geq 0$. The functions which are regular in a point form a ring, $\mathcal{O}_{P_i}$, which has a unique maximal principal ideal:

$$\mathcal{M}_{P_i} = \{f \in \mathbb{F}_q(\chi) \mid \mathbf{v}_{P_i}(f) > 0\} = \langle t_i \rangle$$

where t_i satisfies $\mathbf{v}_{P_i}(t_i) = 1$. t_i is then called a local parameter in P_i. Furthermore, the group of units of $\mathcal{O}_{P_i}$ is given by

$$\mathcal{O}_{P_i} \backslash \mathcal{M}_{P_i} = \{f \in \mathbb{F}_q(\chi) \mid \mathbf{v}_{P_i}(f) = 0\}$$

Any non-zero function, $f \in \mathbb{F}_q(\chi)$, can be written uniquely up to the choice of local parameter as follows:

$$f = t_i^{\mathbf{v}_{P_i}(f)} u_f$$

where u_f is a unit, that is $u_f \in \mathcal{O}_{P_i} \backslash \mathcal{M}_{P_i}$. This will be called the standard representation of f (with respect to the local parameter t_i). More details can be found in [37].

A class of algebraic geometry codes is given by

$$C_{\mathcal{L}}(P, \ell P_\infty) = \{(f(P_1), \ldots, f(P_{n'})) \mid f \in \mathcal{L}(\ell P_\infty)\},\ \ell < n'$$

where $P = \{P_1, \ldots, P_{n'}\}$ and

$$\mathcal{L}(\ell P_\infty) = \{f \in \mathbb{F}_q(\chi) \mid \mathbf{v}_{P_\infty}(f^{-1}) \leq \ell \wedge \mathbf{v}_Q(f) \geq 0 \text{ for all } Q \neq P_\infty\}$$

The length of this code is n', and if g denotes the genus of χ and $2g-1 \leq \ell < n'$ then the dimension of the code is $k' = \ell - g + 1$ and the minimum distance is lower bounded by $d^* = n' - \ell$ since the number of zeroes of a non-zero function cannot exceed the number of poles.

$\mathcal{L}(\ell P_\infty)$ is a vector space over $\mathbb{F}_q$ and for $\ell \geq 2g-1$ the dimension is $\ell - g + 1$. Recall that the non-negative integers, $\mathbb{N}$, are divided into gaps and non-gaps by calling $\ell \in \mathbb{N}$ a gap if and only if $\mathcal{L}(\ell P_\infty) \backslash \mathcal{L}((\ell-1)P_\infty) = \emptyset$. The number of gaps equals the genus, g, of the curve defining the function field. For $\ell \in \mathbb{N}$, let $\mathbf{g}(\ell)$ denote the number of gaps less than or equal to ℓ. That is

$$\mathbf{g}(\ell) := \ell - \dim(\mathcal{L}(\ell P_\infty)) + 1 \tag{6.10}$$

If $\ell \geq 2g-1$ then $\mathbf{g}(\ell) = \ell - (\ell - g + 1) + 1 = g$.

It is well known that $\mathcal{L}(\ell P_\infty)$ has a basis, $\phi_0, \ldots, \phi_{\ell - \mathbf{g}(\ell)}$ where the pole order at P_∞ is increasing:

$$\mathbf{v}_{P_\infty}(\phi_0^{-1}) < \mathbf{v}_{P_\infty}(\phi_1^{-1}) < \cdots < \mathbf{v}_{P_\infty}(\phi_{\ell-\mathbf{g}(\ell)}^{-1}) \tag{6.11}$$

And conversely, any set of $\ell - \mathbf{g}(\ell) + 1$ functions having increasing pole order is a basis of $\mathcal{L}(\ell P_\infty)$. However, the following theorem (from [13]) shows the existence of increasing pole bases where also the zero multiplicity of a given point – different from P_∞ – is increasing for some permutation of the basis functions. Furthermore, the proof of the theorem describes a strategy to find these bases.

Theorem 6.20
Let P_i ($i \in \{1, \ldots, n'\}$) be a point. Then there exist functions,

$$\phi_{0,i}, \ldots, \phi_{\ell - \mathbf{g}(\ell), i}$$

with mutually different pole orders at P_∞ such that

$$\mathcal{L}(\ell P_\infty) = \text{span}\{\phi_{0,i}, \ldots, \phi_{\ell-\mathbf{g}(\ell),i}\}$$

and furthermore,

$$\mathbf{v}_{P_i}(\phi_{0,i}) < \mathbf{v}_{P_i}(\phi_{1,i}) < \cdots < \mathbf{v}_{P_i}(\phi_{\ell-\mathbf{g}(\ell),i})$$

In the following, such a basis will be called an increasing zero basis with respect to the point P_i. □

Proof:
Suppose that some increasing pole basis, $B = \{\phi_0, \ldots, \phi_{\ell-\mathbf{g}(\ell)}\}$, of $\mathcal{L}(\ell P_\infty)$ is given (as in 6.11). Let $B_i := \emptyset$ and do the following:

Let $e := \min\{\mathbf{v}_{P_i}(f) \mid f \in B\}$ and $A := \{f \in B \mid \mathbf{v}_{P_i}(f) = e\}$.

If $|A| = 1$ then let

$$B := B \backslash A$$

$$B_i := B_i \cup A$$

If $|A| > 1$ then let $a_1, \ldots, a_{|A|}$ be an enumeration of the elements in A such that $\mathbf{v}_{P_\infty}(a_1^{-1}) < \mathbf{v}_{P_\infty}(a_j^{-1})$ for $j \neq 1$. Write each a_j as its standard representation:

$$a_j = t_i^e u_j \ , \quad j = 1, \ldots, |A|$$

where $\mathbf{v}_{P_i}(u_j) = 0$ and $\mathbf{v}_{P_i}(t_i) = 1$. Now let

$$B := (B \backslash A) \cup \{u_1(P_i)a_j - u_j(P_i)a_1 \mid j = 2, \ldots, |A|\}$$

$$B_i := B_i \cup \{a_1\}$$

This ensures that $v_{P_i}(f) > e$ for all $f \in B$ and that the pole orders are unchanged.

The process above is repeated $\ell - \mathbf{g}(\ell) + 1$ times until $B = \emptyset$. After this, B_i holds an increasing zero basis of $\mathcal{L}(\ell P_\infty)$ with respect to P_i since B_i is constructed so that two elements cannot have the same valuation in P_i. ■

Notice that $\mathbf{v}_{P_i}(\phi_{0,i}) \geq 0$ so in general $\mathbf{v}_{P_i}(\phi_{j,i}) \geq j$. Furthermore, each $\phi_{j,i}$ is in $\text{span}\{\phi_0, \ldots, \phi_{\ell-\mathbf{g}(\ell)}\}$ and can be written as

$$\phi_{j,i} = \sum_{j'=0}^{\ell-\mathbf{g}(\ell)} \alpha_{j,i,j'}\phi_{j'} \ , \quad \alpha_{j,i,j'} \in \mathbb{F}_q \tag{6.12}$$

Furthermore, notice that the requirement that an increasing zero basis has different pole orders implies that if $\phi_{0,i}, \ldots, \phi_{\ell-\mathbf{g}(\ell),i}$ is an increasing zero basis of $\mathcal{L}(\ell P_\infty)$ then for any $\ell' \leq \ell$ a subset of this increasing zero basis can be used as an increasing zero basis of $\mathcal{L}(\ell' P_\infty)$. This subset will be denoted by

$$\phi_{0,i}^{(\ell')}, \ldots, \phi_{\ell'-\mathbf{g}(\ell'),i}^{(\ell')}$$

Generally, it cannot be assumed that $\phi_{j,i}^{(\ell')} = \phi_{j,i}$ for all $j \leq \ell' - \mathbf{g}(\ell')$ because an increasing zero basis may need to be permuted to have increasing pole orders (see Example 6.21).

Let $f \in \mathcal{L}(\ell P_\infty)$ then f can be written as

$$f = \sum_{j=0}^{\ell-\mathbf{g}(\ell)} f_{j,i}\phi_{j,i}$$

Notice that $\mathbf{v}_{P_i}(f) \geq j'$ if $f_{j,i} = 0$ for all $j < j'$. If $f \in \mathcal{L}(\ell' P_\infty)$ for some $\ell' \leq \ell$ then f can be written as

$$f = \sum_{j=0}^{\ell'-\mathbf{g}(\ell')} f_{j,i}^{(\ell')}\phi_{j,i}^{(\ell')}$$

The following gives an example of some of the concepts introduced above:

Example 6.21
An example of a function field is the so-called Hermitian function field defined by the Hermitian curve over $\mathbb{F}_{q_1^2}$*:*

$$X^{q_1+1} - Y^{q_1} - Y = 0$$

It is well-known that this curve indeed is nonsingular and absolutely irreducible. Furthermore, the curve contains q_1^3 *affine* $\mathbb{F}_{q_1^2}$*-rational points and*

has genus $\frac{q_1(q_1-1)}{2}$. *In this case, the point* P_∞ *corresponds to the (unique) point at infinity on the homogenization of the Hermitian curve.*

Consider the Hermitian function field over $\mathbb{F}_{16}$. *Then* $g = 6$ *and the gaps are* $1, 2, 3, 6, 7, 10$. *Furthermore,* $1, x, y, x^2, xy, y^2$ *is a basis of* $\mathcal{L}(10P_\infty)$ *with pole orders* $0, 4, 5, 8, 9, 10$. *An increasing zero basis of* $\mathcal{L}(10P_\infty)$ *with respect to the point* $(0, 0)$ *is*

$$1, x, x^2, y, xy, y^2$$

The zero orders of these functions are $0, 1, 2, 5, 6, 10$. *An increasing zero basis of* $\mathcal{L}(5P_\infty)$ *is* $1, x, y$. □

Definition 6.22
Let r *be a positive integer and let* k *satisfy* $g \leq k \leq rn' - g$. *Define the following error-correcting code:*

$$C_{P_\infty}(P, r, k) := \{f(P, r) \mid f \in \mathcal{L}(mP_\infty)\}$$

where $P = \{P_1, \ldots, P_{n'}\}$, $m := k + g - 1$, *and*

$$f(P, r) := (f_{0,1}^{(m)}, \ldots, f_{r-1,1}^{(m)}; f_{0,2}^{(m)}, \ldots, f_{r-1,2}^{(m)}; \ldots; f_{0,n'}^{(m)}, \ldots, f_{r-1,n'}^{(m)})$$

□

Notice that this definition differs slightly from the definition in [32]. As illustrated in Example 6.21 there can be "holes" in the zero order sequence of an increasing zero basis. If there is such a hole among the first r functions of the increasing zero basis at a point, P_i, then taking the i'th chunk of the codewords to be the evaluation of the r first coefficients of the Taylor series with respect a local parameter at P_i gives codewords which are always 0 at some position. This is avoided by the use of increasing zero bases.

Just as the codes $C_{\mathcal{L}}(P, \ell P_\infty)$ can be seen as a generalization of Reed-Solomon codes, the codes $C_{P_\infty}(P, r, k)$ can be seen as a generalization of the codes of Definition 6.1. This is reflected in the following where the notation and most of the results on $C(P, r, k)$-codes presented in the previous sections are generalized to $C_{P_\infty}(P, r, k)$-codes.

The following theorems (from [32], Theorem 6) give the length, dimension, and minimum r-distance of the code $C_{P_\infty}(P, r, k)$.

Theorem 6.23
$C_{P_\infty}(P, r, k)$ is a $\mathbb{F}_q$-linear code with length $n := rn'$ and dimension k. □

Proof:
The length is n by construction and the linearity is straightforward. Consider $f \in \mathcal{L}((k+g-1)P_\infty)\backslash\{0\}$. Suppose that $f(P, r)$ is the zero vector. Then

$$\sum_{i=1}^{n'} \mathbf{v}_{P_i}(f) \geq rn' > k + g - 1$$

So the total zero order of f is greater than the pole order, contradicting the assumption that $f \neq 0$. ■

Theorem 6.24
If $u, v \in C_{P_\infty}(P, r, k)$ with $u \neq v$ then $\mathrm{d}_r(u, v) \geq n - k - g + 1$. □

Proof:
Let $f, h \in \mathcal{L}((k+g-1)P_\infty)$ such that $u = f(P, r)$ and $v = h(P, r)$. For each $i \in \{1, \ldots, n'\}$ and $j = 0, \ldots, r-1$

$$(f-h)_{j,i} = f_{j,i} - h_{j,i} = u_{(i-1)r+j} - v_{(i-1)r+j}$$

so $(f-h)_{j,i} = 0$ if $j < \mathrm{s}_r(u, v, i)$ and therefore

$$\mathbf{v}_{P_i}(f-h) \geq \mathrm{s}_r(u, v, i) \tag{6.13}$$

Now, $\mathrm{s}_r(u, v) = \sum_{i=1}^{n'} \mathrm{s}_r(u, v, i) \leq \sum_{i=1}^{n'} \mathbf{v}_{P_i}(f-h)$. Since $f - h \in \mathcal{L}((k+g-1)P_\infty)$ the sum of zero orders is at most $k+g-1$ so $\mathrm{s}_r(u, v) \leq k+g-1$ which implies

$$\mathrm{d}_r(u, v) = n - \mathrm{s}_r(u, v) \geq n - k - g + 1$$

■

Example 6.25
Let ω be a primitive element of $\mathbb{F}_4$ with $\omega^2 + \omega + 1 = 0$. Consider the Hermitian function field over $\mathbb{F}_4$ defined by the curve $X^3 + Y^2 + Y = 0$. The genus is $g = 1$ and the curve contains 8 points:

$$\begin{aligned} P := \{&(0,0), (0,1), (1,\omega), (1,\omega^2), \\ &(\omega,\omega), (\omega,\omega^2), (\omega^2,\omega), (\omega^2,\omega^2)\} \end{aligned}$$

P_∞ corresponds to the (unique) point at infinity on the homogenization of the Hermitian curve. An increasing zero basis of $\mathcal{L}(14P_\infty)$ with respect to the point $P_i = (x_i, y_i)$ for $i \in \{1, \ldots, 8\}$ is given by

$\phi_{0,i} = 1$
$\phi_{1,i} = U$
$\phi_{2,i} = U^2$
$\phi_{3,i} = x_i^2 U + V$
$\phi_{4,i} = x_i^2 U^2 + UV$
$\phi_{5,i} = x_i U + x_i^2 V + U^2 V$
$\phi_{6,i} = x_i U + V^2$
$\phi_{7,i} = x_i^3 U + x_i V + x_i^2 U^2 + x_i V^2 + UV^2$
$\phi_{8,i} = x_i U + x_i^2 V + x_i UV + x_i^2 V^2 + x_i UV^2 + U^2 V^2$
$\phi_{9,i} = x_i^2 U + x_i^3 V + x_i U^2 + x_i^3 V^2 + x_i U^2 V + x_i^2 UV^2 + V^3$
$\phi_{10,i} = x_i^3 U + x_i V + x_i^2 U^2 V + x_i^3 UV^2 + x_i V^3 + x_i^2 U^2 V^2 + UV^3$
$\phi_{11,i} = x_i U + x_i^2 V + x_i^3 U^2 + x_i UV + x_i^3 U^2 V + x_i UV^3 + U^2 V^3$
$\phi_{12,i} = x_i^2 U + x_i^3 V + x_i^2 UV + x_i^3 V^2 + x_i^2 UV^2 + V^4$
$\phi_{13,i} = x_i^2 U^2 + x_i^3 UV + x_i^2 U^2 V + x_i^3 UV^2 + x_i^2 U^2 V^2 + UV^4$

where $U := x - x_i$ and $V = y - y_i$. Consider the code $C_{P_\infty}(P, 2, 11)$. For any i, $\mathcal{L}(11P_\infty) = \mathrm{span}\{\phi_{0,i}, \ldots, \phi_{10,i}\}$ so the function

$$f = \omega + \omega x + \omega y + \omega^2 xy + xy^2 + \omega^2 xy^3$$

is encoded as

$$f(P, 2) = (\omega, \omega, 0, \omega^2, \omega, 0, \omega, 1, 0, \omega^2, 1, \omega, \omega^2, 1, 0, 0)$$

□

6.8 AG decoding

The list decoding algorithm for Reed-Solomon codes in [13] by V. Guruswami and M. Sudan is generalized in the same paper to work for a broad class of algebraic geometry codes. Here the method of section 6.4 will be generalized to a list decoding method for the code $C_{P_\infty}(P, r, k)$.

For $f \in \mathcal{L}((k + g - 1)P_\infty)$ and $u \in F_q^n$ with $n = rn'$ short notations are defined as in (6.4).

Let R denote the following vector space:

$$R := \bigcup_{\ell=0}^{\infty} \mathcal{L}(\ell P_\infty)$$

Suppose that $R = \text{span}\{\phi_\ell \mid \ell \geq 1\}$ with the pole orders of the ϕ_ℓ's being strictly increasing. Then $R[z] = \text{span}\{\phi_\ell z^j \mid \ell \geq 1 \wedge j \geq 0\}$ (where z is transcendental over $\mathbb{F}_q(\chi)$). A total ordering on these basis functions will be defined by associating a non-negative integer – called the weight – to each function. The ordering will be parameterized by the number associated with z. Let this be denoted by $\rho(z)$. Then the weight of the basis function $\phi_\ell z^j$ is given by

$$\rho(\phi_\ell z^j) = \mathbf{v}_{P_\infty}(\phi_\ell^{-1}) + j\rho(z) \tag{6.14}$$

An ordering can now be defined using some lexicographic rule to break ties, for example:

$$\begin{array}{l} \phi_\ell z^j < \phi_a z^b \Leftrightarrow \\ \rho(\phi_\ell z^j) < \rho(\phi_a z^b) \vee (\rho(\phi_\ell z^j) = \rho(\phi_a z^b) \wedge j < b) \end{array} \tag{6.15}$$

However, in this context only the weighting is important.

ρ is extended to any non-zero function in $R[z]$ by the following definition:

Definition 6.26
Let $f \in R[z]\backslash\{0\}$. Suppose that $f = \sum_{\ell,j} f_{\ell,j}\phi_\ell z^j$ and that $\rho(z)$ is given. Then the weight of f is defined as

$$\rho(f) = \max\{\rho(\phi_\ell z^j) \mid f_{\ell,j} \neq 0\}$$

with ρ given by (6.14). □

The following lemma describes the weight of the basis functions:

Lemma 6.27
Suppose that the basis functions, $\psi_0, \psi_1, \ldots$, of $R[z]$ are enumerated increasingly with respect to the ordering induced by the weighting $\rho(z) \geq 2g - 1$:

$$\rho(\psi_0) \leq \rho(\psi_1) \leq \ldots$$

Let $j \in \mathbb{N}$ be given and let b and t satisfy

$$\binom{b}{2}\rho(z) - (b-1)g \leq j < \binom{b+1}{2}\rho(z) - bg$$

$$tb - \mathbf{g}(t) \leq j - (\binom{b}{2}\rho(z) - (b-1)g) < (t+1)b - \mathbf{g}(t+1)$$

where $\mathbf{g}(t)$ is given by (6.10).

The weight of ψ_j is now given by

$$\rho(\psi_j) = (b-1)\rho(z) + t$$

□

Proof:
Group the basis functions into disjoint sets, $M_1, M_2, \ldots$, where

$$M_c = \{\psi_\ell \mid (c-1)\rho(z) \leq \rho(\psi_\ell) < c\rho(z)\}$$

Observe that for each t' with $0 \leq t' < \rho(z)$ the number of functions in M_c with weight $t'+(c-1)\rho(z)$ is exactly c if t' is a non-gap and $c-1$ if t' is a gap. So $|M_c| = c(\rho(z)-g)+(c-1)g = c\rho(z)-g$ and $|M_1|+|M_2|+\ldots+|M_{c-1}| = \binom{c}{2}\rho(z) - (c-1)g$. By the definition of b it is seen that $\psi_j \in M_b$ and by the definition of t, $\rho(\psi_j) = (b-1)\rho(z)+t$. ■

Definition 6.8 is generalized as follows:

Definition 6.28
Let $w = (w_0, \ldots, w_{n-1}) \in \mathbb{F}_q^n$ with $n = rn'$. Then define

$$w^{(i)} = \sum_{j=0}^{r-1} w_{(i-1)r+j}\phi_{j,i}^{(k+g-1)}$$

□

Notice that for any $f \in \mathcal{L}(\ell P_\infty)$,

$$\mathbf{v}_{P_i}(f - w^{(i)}) \geq \mathrm{s}_r(f, w, i)$$

Furthermore, if $Q(z) = \sum_{\alpha=0}^{\deg(Q)} Q^{(\alpha)} z^\alpha \in R[z]$ then Q can be written as follows:

$$Q(z) = \sum_{\alpha=0}^{\deg(Q)} Q^{(\alpha,i)}(z - w^{(i)})^\alpha \ , \quad Q^{(\alpha,i)} \in R$$

Now the algorithm can be stated:

Algorithm 6.29
As input take the code, $C_{P_\infty}(P, r, k)$, a received word, w, and a parameter, $s \geq 1$.

Let $\rho(z) := k + g - 1$ and

$$\ell_s := (b_s - 1)\rho(z) + t$$

where b_s and t satisfy

$$\binom{b_s}{2}\rho(z) - (b_s - 1)g \leq n\binom{s+1}{2} < \binom{b_s+1}{2}\rho(z) - b_s g$$

$$tb_s - \mathbf{g}(t) \leq n\tbinom{s+1}{2} - (\tbinom{b_s}{2}\rho(z) - (b_s - 1)g) < (t+1)b_s - \mathbf{g}(t+1)$$

Determine $Q(z) \in R[z]\backslash\{0\}$ so that

$$\rho(Q) \leq \ell_s$$

and furthermore for $i \in \{1, \ldots, n'\}$, $\alpha \in \{0, \ldots, s-1\}$, and $j \in \{0, \ldots, r(s-\alpha) - 1\}$,

$$Q_{j,i}^{(\alpha,i)} = 0 \tag{6.16}$$

Next, let

$$\tau_s := n - \left\lfloor \frac{\ell_s}{s} \right\rfloor - 1$$

and find all factors of Q of the form $z - f$ with $f \in \mathcal{L}((k + g - 1)P_\infty)$. If $\mathrm{d}_r(f, w) \leq \tau_s$ then include f in the output list. □

To prove that the output list contains all codewords $f(P, r) \in C(P, r, k)$ with $\mathrm{d}_r(f, w) \leq \tau_s$ it must be proven that the polynomial Q exists and that it has the right factors.

Theorem 6.30
$Q(z)$ satisfying the conditions above exists. □

Proof:
Equation 6.16 states $n'(rs^2 - r(0 + 1 + \ldots + (s-1))) = n\tbinom{s+1}{2}$ conditions on the polynomial Q. Each of these conditions is a homogeneous linear equation in the coefficients of Q. By Lemma 6.27 there are at least $n\tbinom{s+1}{2} + 1$ basis functions of $R[z]$ with weight at most ℓ_s, so there are $n\tbinom{s+1}{2} + 1$ unknown coefficients. Therefore, a non-zero solution exists. ■

Lemma 6.31
If $f \in \mathcal{L}((k+g-1)P_\infty)$ *then* $\mathbf{v}_{P_i}(Q(f)) \geq s\mathrm{s}_r(f,w,i)$. □

Proof:

$$Q(f) = \sum_{\alpha=0}^{b_s-1} (f - w^{(i)})^\alpha Q^{(\alpha,i)}$$

Since $\mathbf{v}_{P_i}(f - w^{(i)}) \geq \mathrm{s}_r(f,w,i)$ we have that

$$\mathbf{v}_{P_i}((f - w^{(i)})^\alpha) \geq \alpha \mathrm{s}_r(f,w,i)$$

For $\alpha \in \{0, \ldots, s-1\}$ (6.16) ensures that

$$\mathbf{v}_{P_i}(Q^{(\alpha,i)}) \geq r(s-\alpha)$$

Therefore,

$$\mathbf{v}_{P_i}((f - w^{(i)})^\alpha Q^{(\alpha,i)}) \geq r(s-\alpha) + \alpha \mathrm{s}_r(f,w,i) \geq$$
$$\mathrm{s}_r(f,w,i)(s-\alpha) + \alpha \mathrm{s}_r(f,w,i) = s\mathrm{s}_r(f,w,i)$$

■

Theorem 6.32
If a codeword $f(P,r) \in C_{P_\infty}(P,r,k)$ *has* $\mathrm{d}_r(f,w) \leq \tau_s$ *then* $(z-f)|Q$. □

Proof:
By Lemma 6.31 $\sum_{i=1}^{n'} \mathbf{v}_{P_i}(Q(f)) \geq s\mathrm{s}_r(f,w)$, but

$$\mathrm{d}_r(f,w) \leq n - \left\lfloor \frac{\ell_s}{s} \right\rfloor - 1 \Rightarrow \mathrm{s}_r(f,w) \geq \left\lfloor \frac{\ell_s}{s} \right\rfloor + 1$$

and $\mathbf{v}_{P_\infty}(Q(f)^{-1}) \leq \ell_s < s\mathrm{s}_r(f,w)$. So $Q(f) = 0$ and $z - f$ is therefore a factor of Q. ■

An upper bound on the size of the output list is given by the following theorem:

Theorem 6.33
The number of codewords returned by Algorithm 6.29 is less than b_s. □

Proof:
By the proof of Lemma 6.27 the degree of Q is at most $b_s - 1$. Therefore, Q can have at most $b_s - 1$ factors of the form $z - f$ so the number of codewords returned by the algorithm is at most $b_s - 1$. ■

A simple modification of Algorithm 6.29 (generalizing Algorithm 6.15) gives an efficient and simple algorithm for unique decoding of the code $C_{P_\infty}(P, r, k)$. However, this algorithm which is described in the following, is only guaranteed to correct up to r-distance $\left\lfloor \frac{n-k-g}{2} \right\rfloor - g$, which is g less than half the minimum r-distance. To be guaranteed to correct up to (and beyond) r-distance $\left\lfloor \frac{n-k-g}{2} \right\rfloor$ Algorithm 6.29 must be used for a sufficiently large value of the parameter s.

Let $\alpha+\beta \leq \ell$. For any $a \in \{0, \ldots, \alpha - \mathbf{g}(\alpha)\}$ and $b \in \{0, \ldots, \beta - \mathbf{g}(\beta)\}$, $\phi_{a,i}^{(\alpha)} \phi_{b,i}^{(\beta)} \in \mathcal{L}(\ell P_\infty)$. Define $c(a, b, j)$ by

$$\phi_{a,i}^{(\alpha)} \phi_{b,i}^{(\beta)} = \sum_{j=a+b}^{\ell - \mathbf{g}(\ell)} c(a, b, j) \phi_{j,i}$$

This makes sense since $\mathbf{v}_{P_i}(\phi_{a,i}^{(\alpha)} \phi_{b,i}^{(\beta)}) \geq a + b$. Notice that $c(a, b, j)$ depends on ℓ, α, and β as well, however, in the following these numbers will be given by the context.

Algorithm 6.34
Let $Q = Q^{(0)} + zQ^{(1)} \in \mathbb{F}_q(\chi)[z] \backslash \{0\}$ where z is transcendental over $\mathbb{F}_q(\chi)$, and with

$$Q^{(0)} \in \mathcal{L}((\left\lfloor \tfrac{n+k+g}{2} \right\rfloor + g - 1)P_\infty) \text{ and}$$
$$Q^{(1)} \in \mathcal{L}((\left\lceil \tfrac{n-k-g+2}{2} \right\rceil + g - 1)P_\infty)$$

and furthermore for $1 \leq i \leq n'$ and $j < r$:

$$Q_{j,i}^{(0)} + \sum_{a=0}^{j} \sum_{b=0}^{j-a} c(a, b, j) w_{(i-1)r+a} Q_{b,i}^{(1)} = 0 \tag{6.17}$$

If there exists a codeword $f(P, r) \in C_{P_\infty}(P, r, k)$ with $\mathrm{d}_r(f, w) \leq \left\lfloor \frac{n-k-g}{2} \right\rfloor - g$ then $f = \frac{-Q^{(0)}}{Q^{(1)}}$. □

Notice that for a (n', k') AG-code it is normally possible to correct up to $\left\lfloor \frac{n'-k'-g}{2} \right\rfloor$ errors using a relatively sophisticated algorithm, see for example [34]. If a Welch-Berlekamp type algorithm is used (the above method for $r = 1$) only $\left\lfloor \frac{n'-k'-g}{2} - g \right\rfloor$ errors are guaranteed to be corrected.

For example, consider using a Hermitian code over $\mathbb{F}_{16}$ with length 64 and dimension 48, and compare this to using $C_{P_\infty}(P, 4, 192)$ based on the same curve. 4 codewords in the Hermitian code will be able to correct some error-patterns of weight up to $4\left\lfloor\frac{64-48-6}{2}\right\rfloor = 20$, however, a codeword of $C_{P_\infty}(P, 4, 192)$ will be able to correct some error-patterns with weight up to $\left\lfloor\frac{256-192-6}{2}\right\rfloor - 6 = 23$.

The following example shows the use of Algorithm 6.34:

Example 6.35
This continues Example 6.25. Suppose that $f(P, 2)$ is sent, but the following word is received:

$$w = (\omega, \omega, 0, \omega^2, \omega, 1, \omega, 1, 0, \omega^2, 1, \omega, \omega^2, 1, 0, \omega)$$

which means that 2 errors happened, on position 5 and 15 respectively with the leftmost position being number 0. So w has 2-distance 2 to $f(P, 2)$.

$C_{P_\infty}(P, 2, 11)$ is a $(16, 11)$ code with minimum r-distance 5. In this case the method in the beginning of this section should be able to correct errors only up to 2-distance $\left\lfloor\frac{16-11-1}{2}\right\rfloor - 1 = 1$. But proceeding as described, the polynomial Q is determined as

$$Q = (1 + \omega x + \omega^2 x^2 + y^2 + x^2 y + xy^2 + \omega x^2 y^2 + xy^3 + x^2 y^3 + \omega xy^4) + (\omega^2 + \omega x + \omega^2 y)z$$

and it can be verified that indeed

$$(\omega^2 + \omega x + \omega^2 y)f = 1 + \omega x + \omega^2 x^2 + y^2 + x^2 y + xy^2 + \omega x^2 y^2 + xy^3 + x^2 y^3 + \omega xy^4$$

So the method corrects the two errors in this case. □

Experiments indicate that the algorithm often corrects up to r-distance $\left\lfloor\frac{n-k-g}{2}\right\rfloor$.

Notice that if using a $C_{P_\infty}(P, r, k)$-code in the situation of Example 6.18 the effect will be close to that of using a very long MDS code. For example, the $(256, 192)$-code $C_{P_\infty}(P, 4, 192)$ mentioned above will in practice usually correct up to 29 errors.

6.9 Conclusion

In this paper efficient list-decoding methods have been presented for the codes introduced in [32]. The codes are generalizations of Reed-Solomon and one point algebraic geometry codes. The decoding algorithms are generalizations of decodings algorithms presented in [13] for Reed-Solomon and algebraic geometry codes, and results analogous to the ones obtained in [13] are obtained here with respect to error-correcting capability and upper bounds on the number of codewords in the output.

When comparing the performance of Reed-Solomon and Hermitian codes with the performance of their r-distance counterparts it is clear that the r-distance codes – which are longer – perform better provided that the error-patterns can be assumed to follow the r-distance. If error-patterns are distributed according to the Hamming distance the performance seems to be at the same order of magnitude, but with slower decoding for the r-distance codes. However, a more precise comparison is still to be made.

Chapter 7

Decoding concatenated codes with Sudan's algorithm

R. Refslund Nielsen[1]

Abstract

This paper describes a decoding method for concatenated codes where the outer code is list decoded by Sudan's algorithm and where the inner code is decoded up to half the minimum distance. The list error-correcting capability of the method is calculated and turns out to be on or above half the designed distance of the concatenated code. The results are compared to decoding of concatenated codes by generalized minimum distance decoding. For both methods the description of the (list) error-correcting capability includes error patterns of weight larger than that of the minimal non-correctable pattern. A second list decoding method for concatenated codes where the inner code is list decoded beyond half the minimum distance is proposed and the error-correcting capability is discussed.

Index terms: List decoding, Sudan's algorithm, concatenated codes, generalized minimum distance decoding.

7.1 Introduction

Concatenated codes are often used in practice because the construction gives long codes over a small alphabet and — equally important — decoding can be done efficiently if efficient decoders exist for each of the two component codes. The true minimum distance of a concatenated code is usually difficult to determine, but a lower bound is the product of the minimum distances of the component codes.

[1]Department of Mathematics, Technical University of Denmark, Building 303, DK-2800 Lyngby, Denmark. E-mail: R.R.Nielsen@mat.dtu.dk

The main problem treated in this paper is how many errors a certain decoding method can correct in the worst case. That is determining the greatest integer, w (or a good lower bound) such that all error patterns of weight at most w will be corrected. The number w will be referred to as the error-correcting capability of the decoding method. Possibly, a considerable amount of error patterns of larger weight will be corrected as well. This is analyzed by the introduction of the **badness** of an error pattern which counts how many of the errors are a burden to the decoder. It should be noted that when doing list decoding, an error pattern is said to be corrected when a small list containing the correct codeword has been produced.

Straightforward decoding of concatenated codes only corrects errors up to a quarter of the designed distance. To decode up to half the designed distance it is necessary to take into account decoding information from the inner code when decoding the outer code and besides do several decodings. This method is described in many variants in the literature, but in all cases the basis is to some extend Forney's generalized minimum distance (GMD) decoding [10]. This decoding method based on GMD decoding will be described in this paper and it is shown that when the reliabilities which are parameters of the algorithm are chosen optimally, this method decodes up to half the designed distance, but not beyond that.

Sudan's algorithm [13] makes it possible to list decode Reed-Solomon codes beyond half the minimum distance so one could hope to correct errors beyond half the product bound of concatenated codes if Sudan's algorithm can be used in the decoding. For a special class of codes (Reed-Solomon codes concatenated with Hadamard codes) this is actually done by Guruswami and Sudan in [14].

Sudan's algorithm is described in this paper and improvements are made on the choice of parameters in the algorithm, which reduces the asymptotic running time of list decoding Reed-Solomon codes from $\mathrm{O}(n^7)$ to $\mathrm{O}(n^{2+\gamma})$ for an arbitrarily small positive constant γ when list decoding a fractional number of errors equal to $1 - \sqrt{\kappa}$ where κ is the information rate.

In this paper it is assumed that the outer code in a concatenated code is decodable by Sudan's algorithm, however, for simplicity the analysis will be made only for the case where the outer code is a Reed-Solomon code. The inner code can be an arbitrary linear code. When using the

same inner decoder as the one required for GMD decoding, it is shown that using only a single run of Sudan's algorithm, errors can always be corrected up to half the designed distance and that this is optimal if the minimum distance of the inner code is even or if erasures can occur. If the minimum distance of the inner code is an odd constant, an asymptotic (in a certain sense) bound on the list error-correcting capability is given which is strictly larger than half the designed distance. Furthermore, an algorithm for determining a lower bound on the list error-correcting capability of a given code is described.

The situation where the inner code is list decoded is treated as well. In this case analysis is difficult (since it involves the coset weight distributions of the inner code), but a method to calculate the list decoding capability of the proposed algorithm is found and examples are given where the list decoding capability is well beyond half the designed distance.

The paper is organized as follows: Section II defines some basic notations and Section III describes the construction of concatenated codes. Section IV describes decoding of concatenated codes using GMD decoding and defines the tools of analysis — including the badness of an error pattern — which are used in the rest of the paper. In Section V Reed-Solomon codes and list decoding with Sudan's algorithm are described. Section VI shows how to use Sudan's algorithm in the decoding of concatenated codes up to half the designed distance. In Section VII the error-correcting capability is calculated in the case where the minimum distance of the inner code is odd and examples are given on calculating the error-correcting capability as well as on performing the decoding. Section VIII describes some special cases where an improved error-correcting capability can be obtained by list decoding of the inner code and Section IX contains some concluding remarks.

The Appendix contains proofs of some of the theorems and of the correctness of some of the algorithms of the paper.

7.2 Notation

In this section some common notation for the rest of the paper is defined.

Let $\mathbb{N}$ denote the set of non-negative integers and let $\mathbb{R}$ denote the set of real numbers. For any $x \in \mathbb{R}$ the largest integer smaller than or equal

to x is denoted by $\lfloor x \rfloor$. The smallest integer larger than or equal to x is denoted by $\lceil x \rceil$. For $x \in \mathbb{R}$ and $i \in \mathbb{N}$ it is straightforward to see the following equalities:

$$\begin{aligned} \lfloor x \pm i \rfloor &= \lfloor x \rfloor \pm i, \quad \lceil x \pm i \rceil = \lceil x \rceil \pm i \\ i - \lfloor x \rfloor &= \lceil i - x \rceil, \quad i - \lceil x \rceil = \lfloor i - x \rfloor. \end{aligned}$$

Furthermore, the cardinality of a set, $\mathcal{S}$, is denoted by $|\mathcal{S}|$.

Let $\mathbb{F}_q$ denote a finite field with q elements. For a positive integer n the set of n-tuples over $\mathbb{F}_q$ is denoted by $\mathbb{F}_q^n$. This set is a vector space of dimension n over $\mathbb{F}_q$. Furthermore, the finite field with q^n elements can be seen as a vector space of dimension n over $\mathbb{F}_q$. Therefore, $\mathbb{F}_{q^n}$ and $\mathbb{F}_q^n$ are isomorphic as vector spaces over $\mathbb{F}_q$.

Given a vector v of n elements, the element at position $j \in \{1, \ldots, n\}$ will be denoted by v_j such that

$$v = (v_1, \ldots, v_n).$$

For vectors, $u, v \in \mathbb{F}_q^n$, the **Hamming distance** will be denoted by $\mathrm{d}(u, v)$ and is given by the number of positions where u and v differ:

$$\mathrm{d}(u, v) := |\{j \mid u_j \neq v_j\}|.$$

The **Hamming weight** of a vector u is denoted by $\mathbf{w}(u) := \mathrm{d}(u, 0)$.

A linear (n, k) code over $\mathbb{F}_q$ is a k-dimensional subspace of $\mathbb{F}_q$ and a linear (n, k, d) code is a (n, k) code where the minimal Hamming distance between two different codewords (vectors in the subspace) equals d.

The following extension of the Hamming distance will become useful. Let $\mathcal{A}_q := \mathbb{F}_q \cup \{?\}$ and define the function $\delta_{\mathcal{A}} : \mathcal{A}_q \times \mathcal{A}_q \to \{0, 1/2, 1\}$ by

$$\delta_{\mathcal{A}}(a, b) := \begin{cases} 0 & \text{if } a = b \\ 1/2 & \text{if } a \neq b \text{ and } ? \in \{a, b\} \\ 1 & \text{otherwise.} \end{cases} \tag{7.1}$$

Let $\mathbb{N}/2 = \{0, 1/2, 1, 3/2, \ldots\}$. It can be seen that the function $\mathrm{d}_{\mathcal{A}} : (\mathcal{A}_q^n)^2 \to \mathbb{N}/2$ defined by

$$\mathrm{d}_{\mathcal{A}}(u, v) = \sum_{i=1}^{n} \delta_{\mathcal{A}}(u_i, v_i) \tag{7.2}$$

is a distance function on $\mathcal{A}_q^n$. — Maybe it seems wrong that $\delta_{\mathcal{A}}(?,?)$ equals 0 instead of 1/2, however, this is necessary in order to get a distance function. In any case, this is not important to this paper since at least one of the arguments of $\mathrm{d}_{\mathcal{A}}$ will always be an element of $\mathbb{F}_q^n$. — We also define $\mathbf{w}_{\mathcal{A}}(u) := \mathrm{d}_{\mathcal{A}}(u,0)$. Addition and subtraction will be extended to $\mathcal{A}_q$ by defining the result of an addition or subtraction involving ? to be ?.

The interpretation of the symbol ? above is an erasure. In principle, there are two types of decoding: (I) decoding errors only and (II) decoding errors and erasures. Type I is clearly a special case of type II, so decoding can be analyzed in a unified way by looking only on type II. However, sometimes results can be strengthened or simplified if type I is assumed.

Unless stated otherwise, throughout the paper the number of errors which occurred during transmission will refer to the quantity $\mathrm{d}_{\mathcal{A}}(r,\hat{c})$ where $\hat{c}$ is the sent codeword and r is the received codeword. Therefore, an erasure "counts" as half an error and the number of errors belongs to $\mathbb{N}/2$. The number of errors is also referred to as the weight of the error pattern. It will often be necessary to enumerate all possible error weights. Therefore, for type II decoding, define

$$I := \{0, 1/2, \ldots, n\} \quad \text{(errors/erasures)}.$$

For type I decoding, define I by

$$I := \{0, 1, \ldots, n\} \quad \text{(errors only)}.$$

In either case we also define

$$I_R := \{i \in I \mid i \leq R\}, \quad R \in \mathbb{R}.$$

7.3 Concatenated codes

Let $\mathcal{C}$ be an (n,k) linear code over $\mathbb{F}_q$. By definition $\mathcal{C}$ is a k-dimensional subspace of $\mathbb{F}_q^n$ and therefore isomorphic to $\mathbb{F}_q^k$. Let $\mathrm{enc}_{\mathcal{C}} : \mathbb{F}_q^k \to \mathcal{C}$ be an arbitrary isomorphism. Then $\mathrm{enc}_{\mathcal{C}}$ encodes information (in the form of an element of $\mathbb{F}_{q^k}$) into codewords and the inverse mapping, $\mathrm{enc}_{\mathcal{C}}^{-1}$, extracts the information from a codeword.

Let $\mathcal{C}_1$ be a linear (n,k,d) code over $\mathbb{F}_q$ and let $\mathcal{C}_2$ be a linear (N,K,D) code over $\mathbb{F}_{q^k}$. Then the **concatenation** of $\mathcal{C}_2$ and $\mathcal{C}_1$ is defined as follows:

$$\mathcal{C}_1 * \mathcal{C}_2 := \{(\mathrm{enc}_{\mathcal{C}_1}(u_1), \ldots, \mathrm{enc}_{\mathcal{C}_1}(u_N)) \mid (u_1, \ldots, u_N) \in \mathcal{C}_2\}.$$

In this construction, $\mathcal{C}_1$ is called the **inner** code and $\mathcal{C}_2$ is called the **outer** code. Concatenation was first described in [9] and is now standard text book material. See for example MacWilliams and Sloane [24, Chapter 10, §11].

It is well-known (and straightforward to see) that $\mathcal{C}_1 * \mathcal{C}_2$ is a linear $[Nn, Kk]$ code over $\mathbb{F}_q$. If the minimum distance of $\mathcal{C}_1$ and $\mathcal{C}_2$ is d and D respectively, then the minimum distance $\mathcal{C}_1 * \mathcal{C}_2$ is at least dD since a non-zero codeword of $\mathcal{C}_2$ must have at least D non-zero positions each of which are encoded into a non-zero codeword of $\mathcal{C}_1$ with at least d non-zero positions. The true minimum distance may be larger depending on the exact structure of the codes $\mathcal{C}_1$ and $\mathcal{C}_2$, however, that is not the subject of this paper. Schematically, the code parameters involved in concatenation are

$$(n, k, d) * (N, K, D) \longrightarrow (nN, kK, \geq dD).$$

We will refer to dD as the **designed distance** of the concatenated code.

Decoding all error patterns of weight up to (and not including) $dD/4$ with concatenated codes can be done easily if a decoder is available for the inner and the outer code correcting up to $d/2$ and $D/2$ errors respectively. The idea is to apply the inner decoder to each of the received inner words (blocks of n symbols from $\mathbb{F}_q$), extract the information, and apply the outer decoder to the corresponding N symbols from $\mathbb{F}_q^N$. It is straightforward to see that this method corrects all error patterns of total weight less than $dD/4$ and that in general there exist error patterns of weight $dD/4$ which are not corrected by this method.

In the next section a well-known algorithm correcting up to $dD/2$ errors will be presented in a new notation that prepares for a new algorithm which is presented in the subsequent sections.

7.4 Decoding concatenated codes

In the literature there are various descriptions (Reddy [29], Reddy and Robinson [30], Weldon [39], Blahut [2, Section 10.3]) of essentially the same

method for decoding concatenated (and other composed) codes. Justesen [18] uses the method to decode the construction of asymptotically good codes presented in the same paper. Blokh and Zyablov [3] (also explained by Ericson in [6]) state the method specifically for concatenated codes. However, in all cases the basis of the method is made by Forney [10] with the idea of generalized minimum distance (GMD) decoding. This decoding method was not formulated specifically as a decoding method for concatenated codes, but as it will be described in this section the application to decoding concatenated codes is straightforward.

Let $\mathcal{C}$ be some code in use with minimum distance d and let $r \in \mathcal{A}_q^n$ be a received word. GMD decoding assumes that a decoding method is available for $\mathcal{C}$ which finds the codeword c (if such codeword exists) that satisfies

$$\mathrm{d}_{\mathcal{A}}(r, c) < d/2.$$

Such a codeword must be unique because the restriction of $\mathrm{d}_{\mathcal{A}}$ to $\mathbb{F}_q^n$ is identical to the Hamming distance so the minimum $\mathrm{d}_{\mathcal{A}}$-distance of $\mathcal{C}$ equals the minimum Hamming distance. It will be assumed that the decoder fails if there is no codeword within distance $d/2$ of the received word.

GMD decoding works by assigning some reliability, $z_j \in [0; 1]$, to each position j of the received word (where $[0; 1]$ denotes the closed interval of real numbers between 0 and 1). A position with an erasure (?) is always assigned reliability 0. The positions of the received word are then placed in reliability classes, $J_1, \ldots, J_m \subseteq \{1, \ldots, n\}$, where the positions in a class, J_i, all have the same reliability which we will denote by $z(J_i)$. Furthermore, it is assumed that the reliabilities of the classes are increasing such that $z(J_1) < \cdots < z(J_m)$. The method then first attempts to decode the received word, r. If this is successful (if the result meets the decoding criterion stated below) then the result is given as output, otherwise, erasures are marked on the positions in J_1 and decoding is attempted. If this is not successful then erasures are marked on the positions in J_2 and decoding is attempted. This continues until the decoding is successful or until the word has a total of d or more erasures in which case the decoding fails.

Forney states the following theorem in [10, Theorem 2]:

Theorem 7.1

If it exists then GMD decoding finds the unique codeword, c, satisfying

$$\sum_{i=1}^{m} ((1 - z(J_i))T_i + (1 + z(J_i))F_i) < d$$

where

$$T_i := |\{j \in J_i \mid r_j = c_j\}|$$
$$F_i := |J_i| - T_i$$

such that T_i is the number of positions with reliability $z(J_i)$ where the received word agrees with c, correspondingly, F_i is the number of positions with reliability $z(J_i)$ where the received word do not agree with c. □

Notice that in the case where the received word has erasures, this theorem can still be used since erasures are always given reliability 0 (so the positions with erasures belong to J_1). In this case we may set $T_1 := 0$ and $F_1 := |J_1|$.

GMD decoding is related to decoding of concatenated codes by the observation that the distance between the received inner word at some position and the output of the inner decoder at that position is a measure of the reliability of the decoded codeword. This will be explained in the following.

Consider a concatenated code, $\mathcal{C}_1 * \mathcal{C}_2$, where $\mathcal{C}_1$ is a (n, k) code over $\mathbb{F}_q$ with minimum distance d and $\mathcal{C}_2$ is a (N, K) code over $\mathbb{F}_{q^k}$ with minimum distance D.

Throughout most of the paper the following assumption will be made on the specification of the inner decoder:

Assumption 7.2 (Inner decoder)
Let $\mathcal{C}_1$ be the inner code in a concatenated code as described above and let $r_j \in \mathcal{A}_q^n$ be a received word. If there exists a codeword, v_j, such that $\mathrm{d}_A(r_j, v_j) < d/2$ then the output of the inner decoder is v_j. In all other cases the inner decoder outputs $(?, \ldots, ?)$ and we will say that the inner decoder failed. □

In particular, this means that decoding on or beyond half the minimum distance of the inner code will not be attempted.

There a several objects involved in the decoding of concatenated codes. The notation which will be used in the rest of the paper is summarized below — a hat ($\hat{}$) means that an object is not known to the receiver:

- Sent codeword: $\hat{c} \in \mathcal{C}_1 * \mathcal{C}_2$.
- Received word: $r \in (\mathcal{A}_q^n)^N$.
- Error pattern: $\hat{e} := \hat{c} - r$.
- Result of inner decodings: $v \in (\mathcal{C}_1 \cup (?, \ldots, ?))^N$.
- Outer received word: $y := (\mathrm{enc}_{\mathcal{C}_1}^{-1}(v_1), \ldots, \mathrm{enc}_{\mathcal{C}_1}^{-1}(v_N))$ where

$$\mathrm{enc}_{\mathcal{C}_1}^{-1}(?, \ldots, ?) = ?.$$

Furthermore, the following definitions are made:

- Observed error weight at position $j \in \{1, \ldots, N\}$:

$$w_j := \begin{cases} \mathrm{d}_{\mathcal{A}}(r_j, v_j) & \text{if } v_j \in \mathcal{C}_1 \\ d/2 & \text{if } v_j = (?, \ldots, ?) \end{cases}$$

  By Assumption 7.2 we have that $0 \leq w_j \leq d/2$ or, more precisely, $w_j \in I_{d/2}$.

- Number of positions with observed error weight $i \in I_{d/2}$:

$$W_i := |\{j \mid w_j = i\}|.$$

- True error weight at position $j \in \{1, \ldots, N\}$:

$$\hat{w}_j := \mathrm{d}_{\mathcal{A}}(r_j, \hat{c}_j).$$

- Number of positions with true error weight $i \in I$:

$$\hat{W}_i := |\{j \mid \hat{w}_j = i\}|.$$

- The true error weight, $\hat{W} := \sum_{j=1}^{N} \hat{w}_j$.

Notice that if v_j is not the correct inner codeword then at the number of errors which have occurred in the j'th inner word must be at least $d - w_j$ (remember that an erasure counts as half an error). A measure of the reliability of the symbol y_j is, therefore, $d/2 - w_j$, and since the reliability in GMD decoding must be between 0 and 1, we normalize by dividing by $d/2$ giving a reliability of

$$z_j := 1 - \frac{2w_j}{d}. \tag{7.3}$$

At this moment there is no reason why this choice of reliabilities should be good, but it will be shown below that they are optimal in the sense that they maximize the weight of the smallest non-correctable error pattern.

Notice that W_i is the number of positions with reliability $1 - 2i/d$ and that the positions where $\hat{w}_j < d/2$ are exactly the positions where v_j agrees with $\hat{c}_j$.

A concatenated code is typically able to decode many error patterns of weight on and beyond half the designed minimum distance of the code. Some of these error patterns are correctable because of special properties of the component codes and some are correctable because of the construction of concatenation. Only the latter case is treated here with the introduction below of the **badness** of an error pattern. The idea is to show that GMD decoding can decode error patterns which have sufficiently low badness.

Definition 7.3 (Badness of error pattern)
*The **badness** of a received inner word, $r_j \in \mathcal{A}_q^n$, is defined as*

$$\hat{b}_j := \begin{cases} w_j & \text{for } \hat{w}_j < d/2 \\ d - w_j & \text{for } \hat{w}_j \geq d/2. \end{cases}$$

The total badness of a received word, $r \in (\mathcal{A}_q^n)^N$, is then

$$\hat{B} := \sum_{j=1}^{N} \hat{b}_j$$

□

Notice that the badness is at most the weight of the error pattern (so $\hat{B} \leq \hat{W}$). Consequently, an error pattern whose badness equals its weight will be called a **worst case** error pattern.

Intuitively, this definition of badness is related to the point of view that a bad error pattern is one which takes the sent codeword directly in direction of another codeword. Informally, $\hat{w}_j - \hat{b}_j$ is the number of errors in the j'th inner word which will not be a burden to the overall decoder.

The following theorem describes the decoding capability of GMD decoding when used on a concatenated code:

Theorem 7.4
GMD decoding with reliabilities given by Eqn. 7.3 on a concatenated code successfully recovers the correct codeword if

$$\hat{B} < dD/2$$

where $\hat{B}$ is the badness of the error pattern as defined in Def. 7.3 and dD is the designed minimum distance of the concatenated code. □

Since $\hat{B} \leq \hat{W}$ we have the following corollary:

Corollary 7.5
GMD decoding with reliabilities given by Eqn. 7.3 on a concatenated code successfully recovers the correct codeword if the weight of the error pattern is less than $dD/2$. □

Proof of the theorem:
Let $\hat{B}_i$ denote the number of positions with badness $i \in I_d$:

$$\hat{B}_i := |\{j \mid \hat{b}_j = i\}|.$$

Then

$$W_i = \begin{cases} \hat{B}_i + \hat{B}_{d-i} & \text{for } i \in I_t \text{ with } t := (d-1)/2 \\ \hat{B}_i & \text{for } i = d/2 \end{cases}$$

and the number of correct and wrong inner codewords with reliability $1 - 2i/d$ is $\hat{B}_i$ and $\hat{B}_{d-i}$ respectively for $i \in I_t$. Furthermore,

$$\hat{B} = \sum_{i \in I_d} i\hat{B}_i.$$

Theorem 7.1 now gives that the correct codeword will be found by GMD decoding if

$$\hat{B}_{d/2} + \sum_{i \in I_t} ((1 - (1 - 2i/d))\hat{B}_i + (1 + (1 - 2i/d))\hat{B}_{d-i}) < D.$$

The left hand side equals

$$\hat{B}_{d/2} + \sum_{i \in I_t} (2i/d)\hat{B}_i + 2(d-i)/d\hat{B}_{d-i}) = 2/d\hat{B}$$

and the theorem follows. ■

On the other hand, we have the following theorem:

Theorem 7.6
For any concatenated code, an error pattern exists that has weight $\lceil dD/2 \rceil$ (half the designed minimum distance) and for which GMD decoding fails for any choice of reliabilities. □

Proof:
If d is even and a worst case error pattern occurs with $\hat{W}_0 = N - D$ and $\hat{W}_{d/2} = D$ then there will be D erasures in the outer word. This will cause the outer decoder to fail for any choice of reliabilities. Such an error pattern exists since the inner code is linear so any inner codeword has another codeword at distance d.

If d is odd, consider the case where D is even (the case where D is odd can be treated similarly). Suppose that a worst case error pattern occurs with $\hat{W}_0 = N - D$ and $\hat{W}_t = \hat{W}_{t+1} = D/2$ where $t := (d-1)/2$ (such an error pattern exists because the inner code is linear). This implies that $W_0 = N - D$ and $W_t = D$ giving two reliability classes. Independently of the choice of reliabilities, there are only two possibilities:

- Erasing no symbols, which gives $D/2$ errors and (in general) decoding fails.
- Erasing the symbols with $w_j = t$ gives D erasures and decoding fails again.

Therefore, for any values of d and D there exists an error pattern with $\lceil Dd/2 \rceil$ errors which will not be corrected by GMD decoding. ■

Consequently, the reliabilities in Eqn. 7.3 are optimal in the sense that they maximize the weight of the smallest non-correctable error pattern — both for erasure-and-error decoding and for error only decoding.

For completeness it should be mentioned that the choice of reliabilities in Eqn. 7.3 is not very special in the sense that it makes the decoding trials in GMD decoding find the correct codeword if $\hat{B} < dD/2$. Any function which

is strictly decreasing for increasing w_j could be used. However, this specific choice gives the correct criterion — that is the correct generalized minimum distance — in Theorem 7.1 for the correct codeword to be identified.

7.5 Sudan's algorithm

In [38] Sudan presented an efficient algorithm for list decoding of Reed-Solomon codes capable of finding a list of all codewords within a distance from the received word which is asymptotically (with the code length) strictly greater than half the minimum distance. This had not been achieved before and it triggered a lot of research. Shokrollahi and Wasserman [35] generalized the algorithm to decode algebraic geometry codes. In [13] Guruswami and Sudan presented an improved algorithm which has a greater error-correcting capability both for Reed-Solomon codes and for algebraic geometry codes. The paper also contains many interesting applications to coding theory, and several ideas for variations of the algorithm. In Algorithm 6.9 and Algorithm 6.29 the algorithm was generalized to handle Reed-Solomon and algebraic geometry codes for the m-metric as introduced by Rosenbloom and Tsfasman in [32].

Furthermore, much research has been done in order to implement the algorithm efficiently. See [11], [33], [41], Chapter 3, and Chapter 4. In [19, Theorem 4] Justesen and Høholdt show that the list decoding capability of the improved version of Sudan's algorithm for Reed-Solomon codes obtains the greatest error-correcting capability possible for MDS codes when the list size is bounded by a constant.

In this section the extension of Sudan's improved algorithm introduced in [13] as Weighted Polynomial Reconstruction is described in the setup of Chapter 3. However, first some basic definitions are needed.

Reed-Solomon codes will here be seen as evaluation codes constructed in the following way:

Definition 7.7 (Reed-Solomon codes)
Let $P := \{P_1, \ldots, P_N\} \subseteq \mathbb{F}_q$ *with* $|P| = N$. *For a given integer,* K, *with* $1 \leq K \leq N$ *the set*

$$\mathrm{RS}(P, K) := \{f(P) \mid f \in \mathbb{F}_q[x] \wedge \deg(f) < K\}$$

where $f(P) := (f(P_1), \ldots, f(P_N))$ *is called a* ***Reed-Solomon code***[2]. $\square$

It is well-known that $\mathrm{RS}(P, K)$ is a linear (N, K) code with minimum distance $D := N - K + 1$. Notice that there is a one-to-one correspondence (actually, a vector space isomorphism) between polynomials of degree less than K and codewords in $\mathrm{RS}(P, K)$.

Furthermore, the concepts of weighted degree and multiplicity of a zero for bivariate polynomials are needed. Let $Q(x, y) \in \mathbb{F}_q[x, y] \backslash \{0\}$ be a bivariate polynomial and define $Q_{\alpha,\beta}$ to be the coefficient of $x^\alpha y^\beta$. That is

$$Q(x, y) = \sum_{\alpha,\beta} Q_{\alpha,\beta} x^\alpha y^\beta, \quad Q_{\alpha,\beta} \in \mathbb{F}_q.$$

Then the (a, b)-**weighted degree** of Q is given by

$$\deg^{(a,b)}(Q) := \max\{a\alpha + b\beta \mid Q_{\alpha,\beta} \neq 0\}.$$

Let $(x_0, y_0) \in \mathbb{F}_q^2$ and consider a bivariate polynomial, Q, as above. The shifted polynomial with respect to (x_0, y_0) will then be defined as

$$Q^{(0)}(x, y) := Q(x + x_0, y + y_0).$$

This implies that

$$Q(x, y) = \sum_{\alpha,\beta} Q^{(0)}_{\alpha,\beta} (x - x_0)^\alpha (y - y_0)^\beta \tag{7.4}$$

and (x_0, y_0) is a **zero of multiplicity** s of Q if

$$\min\{\alpha + \beta \mid Q^{(0)}_{\alpha,\beta} \neq 0\} = s.$$

It can be seen by direct calculation that the coefficients of $Q^{(0)}$ is a linear combination of the coefficients of Q so the condition that (x_0, y_0) is a zero of multiplicity at least s of Q can be written as a system of homogeneous linear equations in the coefficients of Q. The number of equations needed is

$$\sum_{j=0}^{s} j = \binom{s+1}{2}. \tag{7.5}$$

The following lemma is also proved in Lemma 3.4 (and is a special case of Lemma 4.3).

[2]This definition of Reed-Solomon codes is the one described in the original paper by Reed and Solomon [31], however, in some texts Reed-Solomon codes are cyclic of length $q - 1$ and the codes described in this paper would then include extended and/or punctured Reed-Solomon codes.

Lemma 7.8
Let $m_1, m_2, \ldots$ be all the monomials of $\mathbb{F}_q[x, y]$ enumerated such that

$$\deg^{(1,K-1)}(m_1) \leq \deg^{(1,K-1)}(m_2) \leq \ldots$$

for a given positive integer K. Then for any $i \geq 1$,

$$\deg^{(1,K-1)}(m_i) = \left\lfloor \frac{i-1}{\lambda} + \frac{(\lambda-1)(K-1)}{2} \right\rfloor$$

where λ is the integer satisfying

$$\binom{\lambda}{2} \leq \frac{i-1}{K-1} < \binom{\lambda+1}{2}.$$

□

Now we can describe Sudan's algorithm:

Algorithm 7.9
Input: $\mathrm{RS}(P, K)$, *pairs* $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ *where* $(x_j, y_j) \in \mathbb{F}_q \times \mathbb{F}_q$, *and weights* $s_1, \ldots, s_m \in \mathbb{N}$.
Output: $\{f(P) \in \mathrm{RS}(P, K) \mid Z(f) > \ell\}$ *where* ℓ *is given by Eqn. 7.8 below and*

$$Z(f) := \sum_{j, f(x_j)=y_j} s_j. \tag{7.6}$$

- *Let*

$$C := \sum_{j=1}^{m} \binom{s_j+1}{2} \tag{7.7}$$

and

$$\ell := \left\lfloor \frac{C}{\lambda} + \frac{(\lambda-1)(K-1)}{2} \right\rfloor \tag{7.8}$$

where λ is the integer satisfying

$$\binom{\lambda}{2} \leq \frac{C}{K-1} < \binom{\lambda+1}{2} \tag{7.9}$$

- *Find* $Q(x,y) \in \mathbb{F}_q[x,y]\backslash\{0\}$ *such that*
  1. *For each* $j \in \{1,\ldots,m\}$, (x_j, y_j) *is a zero of multiplicity at least* s_j *of* Q.
  2. $\deg^{(1,K-1)}(Q) \leq \ell$ *where* ℓ *is given by Eqn. 7.8.*
- *Find factors of* Q *in the form* $y - f(x)$ *where* $f(P) \in \mathrm{RS}(P,K)$. *If* $Z(f) > \ell$ *then include* $f(P)$ *in the output set.*

□

By condition 1 and Eqn. 7.5 the coefficients of Q have to satisfy a total of C linear equations and Lemma 7.8 then gives that the choice of ℓ made in the algorithm ($\ell := \deg^{(1,K-1)}(m_{C+1})$) is the smallest possible in order to guarantee that Q has at least $C+1$ unknown coefficients such that Q always exists.

The following theorem shows that the algorithm gives the promised output and — which is equally important — that the size of the output has a strong upper bound.

Theorem 7.10
Algorithm 7.9 outputs exactly those codewords, $f(P)$, *where* $Z(f) > \ell$ *with* $Z(f)$ *and* ℓ *being defined by Eqn. 7.6 and Eqn. 7.8 respectively. Furthermore, the output contains at most* $\lambda - 1$ *codewords where* λ *is defined in Eqn. 7.9.* □

Proof:
Consider a pair (x_j, y_j) in the input of the algorithm and some polynomial, $f \in \mathbb{F}_q[x]$. Then Eqn. 7.4 gives

$$Q(x, f(x)) = \sum_{\alpha,\beta} Q^{(j)}_{\alpha,\beta}(x - x_j)^\alpha (f(x) - y_j)^\beta.$$

Suppose that $f(x_j) = y_j$. Then $x - x_j$ divides $f(x) - y_j$ and since (x_j, y_j) is a zero of multiplicity at least s_j of Q this gives that $(x - x_j)^{s_j}$ divides $Q(x, f(x))$ such that x_j is a zero of multiplicity at least s_j of $Q(x, f(x))$. Therefore, the number of zeroes (counted with multiplicity) of $Q(x, f(x))$ is at least $Z(f)$.

Notice that $\deg(Q(x, f(x))) \leq \deg^{(1,K-1)}(Q(x,y)) \leq \ell$ so if $Z(f) > \ell$ then $Q(x, f(x)) = 0$ and $y - f(x)$ is a factor of $Q(x,y)$ so $f(P)$ will be in

the output of the algorithm. By the selection criterion in the last step of the algorithm, codewords with $Z(f) \leq \ell$ will not be in the output.

Furthermore, since the degree in y of Q is at most $\lambda - 1$ there can be at most $\lambda - 1$ factors in the form $y - f(x)$ and, therefore, the output of the algorithm contains at most $\lambda - 1$ codewords. ■

If $\hat{f}(P)$ is the sent codeword then let

$$\hat{Z} := Z(\hat{f}).$$

The goal in Algorithm 7.9 is then to ensure that $\hat{Z} > \ell$ in order to ensure that the correct codeword is among the output. In that situation we will say that Algorithm 7.9 corrects the occurred errors even though the result may not be unique. Furthermore, if $\hat{Z} \leq \ell$ then we will say that Algorithm 7.9 does not correct the occurred errors even though in a given situation the polynomial of the correct codeword may actually be found in the factors of Q.

As mentioned, an efficient implementation of Algorithm 7.9 is a field of active research and will not be discussed in detail here. However, combining Algorithm 5.10 and the factorization method described [11], it seems reasonable to assume a complexity of $\mathrm{O}(\lambda C^2) = \mathrm{O}(C^{5/2}K^{-1/2})$ (informally, the notation $\mathrm{O}(*)$ means "of the same order of magnitude as $*$"). By the dependence of C on the weights, s_j, given as input to Algorithm 7.9 it is seen that the complexity of the algorithm depends dramatically on the choice of weights. Therefore, it is important to analyse how the error-correcting capability of the Algorithm depends on the choice of weights. In the following we will consider the case of Reed-Solomon codes.

The exact $(1, K-1)$-weighted degree of monomials in $\mathbb{F}_q[x, y]$ is given by Lemma 7.8, however, sometimes the following lemma is more convenient.

Lemma 7.11
Let $K > 1$ be a given integer.

1. *For $\ell \in \mathbb{N}$ let $M(\ell)$ denote the set of monomials of $\mathbb{F}_q[x, y]$ of $(1, K-1)$ weighted degree at most ℓ. That is $M(\ell) := \{x^\alpha y^\beta \mid \alpha + (K-1)\beta \leq \ell\}$. Then*

$$|M(\ell)| \geq \frac{(\ell+1)(\ell+K)}{2(K-1)}.$$

2. *Let $m_1, m_2, \ldots$ be the monomials of $\mathbb{F}_q[x,y]$ enumerated increasingly w.r.t. the $(1, K-1)$ weighted degree. That is*

$$\deg^{(1,K-1)}(m_1) \leq \deg^{(1,K-1)}(m_2) \leq \ldots .$$

Then for any $i \in \mathbb{N}$,

$$\left\lceil \sqrt{2i(K-1)} - \frac{K-1}{2} - 1 \right\rceil \leq \deg^{(1,K-1)}(m_i) \leq \left\lceil \sqrt{\frac{(K-1)^2}{4} + 2i(K-1)} - \frac{K-1}{2} - 1 \right\rceil .$$

□

Proof:

1. For convenience, let $b := K-1$. The number of monomials in $M(\ell)$ with degree j in y is $\max\{0, \ell+1-bj\}$, so

$$\begin{aligned} |M(\ell)| &= \sum_{j=0}^{\lfloor (\ell+1)/b \rfloor} \ell + 1 - bj \\ &= (\ell+1)\left\lfloor \frac{\ell+1}{b} + 1 \right\rfloor - \frac{b}{2}\left\lfloor \frac{\ell+1}{b} \right\rfloor \left\lfloor \frac{\ell+1}{b} + 1 \right\rfloor . \end{aligned}$$

Let $\gamma := (\ell+1)/b - \lfloor (\ell+1)/b \rfloor$. Then

$$|M(\ell)| = \frac{1}{2}\left(\frac{(\ell+1)^2}{b} + \ell + 1 + b\gamma(1-\gamma) \right)$$

and the first part of the lemma follows since $\gamma(1-\gamma) \geq 0$.

2. Notice that $\deg^{(1,b)}(m_i) = \min\{\ell \mid i \leq |M(\ell)|\}$ and

$$|M(\ell)| \geq i \Leftrightarrow \frac{(\ell+1)^2}{b} + \ell + 1 + b\gamma(1-\gamma) - 2i \geq 0.$$

Since $\ell \geq 0$ this is true if and only if

$$\ell + 1 \geq \frac{b}{2}\left(\sqrt{1 - 4\gamma(1-\gamma) + 8i/b} - 1 \right).$$

Therefore,

$$\deg^{(1,b)}(m_i) = \left\lceil \frac{b}{2}\sqrt{(1-2\gamma)^2 + \frac{8i}{b}} - \frac{b}{2} - 1 \right\rceil .$$

The second part of the lemma now follows since $(1-2\gamma)^2 \in [0;1]$.

■

In the rest of this section it will be assumed that no erasures occur. Or put in another way, in case of erasures it will be assumed that the code is punctured by removing the erasure positions, after which decoding is done on the punctured code.

Let $RS(P,K)$ be an (N,K) Reed-Solomon code over $\mathbb{F}_q$ and let $y = (y_1,\ldots,y_N) \in \mathbb{F}_q^N$ be a received word. Consider decoding with Algorithm 7.9 such that the input pairs are $(P_1,y_1),\ldots,(P_N,y_N)$ and $s_j = \sigma$ for all $j = 1,\ldots,N$ and for some given positive integer, σ. The error-correcting capability of Algorithm 7.9 in this situation depends on the choice of σ. We have the following theorem which improves [13, Sec. II.B]:

Theorem 7.12
Let the common multiplicity, σ, be chosen such that

$$\sigma = \left\lfloor \frac{U^2 - U(K-1)}{U^2 - N(K-1)} \right\rfloor \tag{7.10}$$

for some integer $U < \sqrt{N(K-1)}$. Then Algorithm 7.9 corrects all error patterns of weight at most $N-U$. □

Proof:
In this situation we have

$$C = \frac{\sigma(\sigma+1)}{2}N \text{ and } \hat{Z} = \sigma(N - \hat{W})$$

where $\hat{W}$ is the weight of the error pattern. Suppose that $\hat{W} \le N-U$. This gives $\hat{Z} \ge \sigma U$ and thus the correct codeword is in the output if $\ell < \sigma U$. Let $\ell' := \sigma U - 1$.

Now consider Lemma 7.11.1 and notice that

$$\begin{aligned} & \frac{(\ell'+1)(\ell'+K)}{2(K-1)} > \frac{\sigma(\sigma+1)}{2}N \\ \Leftrightarrow\ & U(\sigma U + (K-1)) > (\sigma+1)N(K-1) \\ \Leftrightarrow\ & \sigma > \frac{(N-U)(K-1)}{U^2 - N(K-1)} \end{aligned}$$

which is satisfied by the choice of σ in Eqn. 7.10 since

$$\frac{(N-U)(K-1)}{U^2-N(K-1)}+1=\frac{U^2-U(K-1)}{U^2-N(K-1)}.$$

Therefore, $|M(\ell')|>C$. However, the choice of ℓ made in Algorithm 7.9 is the smallest value satisfying $|M(\ell)|>C$. So $\ell\leq\ell'<\sigma U$ and the correct codeword is in the output. ■

Notice that the largest error correcting capability is obtained for $U=\lfloor\sqrt{N(K-1)}\rfloor+1$ in the lemma above. In that case we have $\sigma\leq U^2-U(K-1)$ which is of the same order of magnitude as N^2.

However, letting $U=\lfloor\sqrt{N(K-1)}+3/2\rfloor$ the value of σ satisfies $\sigma\leq(U^2-U(K-1))/\sqrt{N(K-1)}$ which is of the same order of magnitude as N.

Trying to reduce the order of magnitude of σ further results in a loss of error-correcting capability that increases with the code length. However, we may also consider the asymptotic error correcting capability of Algorithm 7.9. Asymptotic in this context means that we consider an infinite sequence of Reed-Solomon codes with parameters $(N_1,K_1),(N_2,K_2),\ldots$, where $N_a\to\infty$ as $a\to\infty$ and K_a/N_a tends to a constant, κ. For each code we can obtain some bound E_a on the error-correcting capability of Algorithm 7.9 such that all error patterns of weight less than E_a will be corrected. The asymptotic fractional error-correcting capability is then the constant ϵ such that $E_a/N_a\to\epsilon$ as $a\to\infty$.

It should be noted that in the case of Reed-Solomon codes, the alphabet size must tend to infinity for a sequence of codes as described above.

The asymptotic error correcting capability of Algorithm 7.9 is given by the following theorem:

Theorem 7.13

1. *Let $\gamma>0$ be an arbitrary (small) positive constant. For $\sigma=O(N^\gamma)$ the asymptotic fractional error-correcting capability of Algorithm 7.9 is $1-\sqrt{\kappa}$.*

2. *For any choice of σ the asymptotic fractional error-correcting capability of Algorithm 7.9 is at most $1-\sqrt{\kappa}$.*

□

Proof:

1. For a given Reed-Solomon code with parameters (N_a, K_a) let

$$U_a := \lfloor \sqrt{N_a K_a (1 + N_a^{-\gamma})} \rfloor.$$

Then $U_a/N_a \to \sqrt{\kappa}$ for $a \to \infty$ and the corresponding value of σ_a given by Eqn. 7.10 satisfies

$$\frac{\sigma_a}{N_a^\gamma} = N_a^{-\gamma} + \frac{(N_a - U_a)(K_a - 1)}{N_a^\gamma (U_a^2 - N_a(K_a - 1))} \to 1 - \sqrt{\kappa}$$

for $a \to \infty$.

2. Suppose that Algorithm 7.9 is used on a given Reed-Solomon code with parameters (N_a, K_a) and on some received word with error weight $\hat{W}_a$. Let ℓ_a be given by Eqn. 7.8 such that $\ell_a = \deg^{(1,K_a-1)}(m_{C_a+1})$ with $C_a = \binom{\sigma_a+1}{2} N_a$. The correct codeword is in the output if and only if $\sigma_a(N_a - \hat{W}_a) > \ell_a$ or, equivalently, $\hat{W}_a < E_a$ with $E_a := N_a - \ell_a/\sigma_a$. Essentially, there are 2 ways to choose σ_a: (1) such that $\sigma_a \to \infty$ for $a \to \infty$ or (2) such that $\sigma_a \to \sigma$ for $a \to \infty$ where $\sigma \in \mathbb{R}_+$. In both cases the lower bound in Lemma 7.11.2 gives that

$$\frac{\ell_a}{N_a \sigma_a} \to \gamma \quad \text{for } a \to \infty$$

for some $\gamma \geq \sqrt{\kappa}$. Therefore, the fractional error correcting capability, $E_a/N_a \to 1 - \gamma \leq 1 - \sqrt{\kappa}$.

■

7.6 Decoding concatenated codes with Sudan's algorithm

In this section Algorithm 7.9 will be used in the decoding of a concatenated code where the inner code is arbitrary and the outer code is a Reed-Solomon code (the generalization to algebraic geometry codes — and possibly other Sudan-decodable codes — is straightforward, but requires a heavier notation which would conceal the general idea, so that is not done here).

Actually, we may even have varying inner codes as long as they have the same minimum distance or it is acceptable to decode the inner codes up to a common distance. Especially, Justesen codes [18] will be included even though no improvement in the error-correcting capability is achieved in that case.

In [14] Guruswami and Sudan use Algorithm 7.9 in list decoding of a special class of concatenated codes with a Hadamard code as the inner code and a Reed-Solomon code as the outer code. Their results rely heavily on the special structure of the inner codes and it is not easy to see the implications for general inner codes. On the other hand, they are able to exploit the special structure of Hadamard codes to obtain a list error-correcting capability which appears to be very good. See also Example 7.31.

In [21] Kötter and Vardy use Algorithm 7.9 in a setting where soft-decision information is available (this means the value of the received word at a given position is seen as a random variable with a known distribution on the code alphabet). However, a connection between their results and the results of this paper is not clear.

In this section it will be assumed that the inner decoder works as specified in Assumption 7.2 (the inner code is decoded only up to half the minimum distance). Under this assumption we are able to state some quite strong upper bounds on the maximal list decoding capability of *any* decoding algorithm:

Theorem 7.14
Let a concatenated code be given where the inner code has minimum distance d and the outer code is a Reed-Solomon code of minimum distance D. Suppose that an inner decoder as specified in Assumption 7.2 is used. Independently of the choice of outer decoder we then have the following:

1. *If erasures can occur in the inner words or if d is even then there exists an error pattern of weight $dD/2$ which cannot be list decoded.*

2. *If d is odd then there exists an error pattern of weight*

$$\frac{d+1}{d}\frac{dD}{2}$$

which cannot be list decoded.

□

Proof:
Let K be the dimension of the outer code. If the outer decoder gets as input a word with D or more errors or erasures (just here, an erasure count as 1) then there are at most $K-1$ correct symbols and at least one information symbol is lost and cannot be recovered by list decoding (asymptotically, if there is a fraction of erasures/errors above $1-\kappa$ then a list of nearest codewords including the correct codeword has a size which is exponential in the code length).

In the case where erasures can occur or where d is even suppose that an error pattern occurs with $d/2$ errors in D inner words. This gives an error pattern of weight $dD/2$ and by the assumption on the inner decoder there will be D erasures in the outer word so the error pattern cannot be list decoded.

If d is odd and erasures cannot occur then D erasures cannot be obtained as above, however, $(d+1)/2$ errors may occur in D inner words giving D erasures or errors in the outer word. This is an error pattern of weight $(d+1)D/2$ that cannot be corrected by list decoding. ■

Notice that the theorem says that when erasures can occur or when d is even then the list decoding capability of a concatenated code under Assumption 7.2 is the same as the capability for unique decoding. This is because of the fact that an MDS code cannot recover from a number of erasures on or beyond the minimum distance.

For odd d the theorem leaves room for some improvement (which, however, vanishes for d tending to infinity).

The following algorithm shows how to combine an inner decoder as specified in Assumption 7.2 with the list decoding method for Reed-Solomon codes in Algorithm 7.9 into a list decoding method for concatenated codes.

Algorithm 7.15

1. *Use an inner decoder as specified in Assumption 7.2 on each component of the received word, r, such that an outer word, $y \in \mathcal{A}_{q^k}^N$, is obtained.*

2. *Use Algorithm 7.9 with the following input: Let $N' := |\{j \mid y_j = ?\}|$. Then $m := N - N'$ and the interpolation points are (P_j, y_j) for $y_j \neq ?$.*

The multiplicities s_j are chosen in a suitable way depending on the reliability of the corresponding symbols y_j (see below).

□

Notice in Algorithm 7.9 that if the zero multiplicity, s_j, is raised for some position, j, then ℓ – the maximal weighted degree of Q – is raised. Let $\hat{f}(P)$ be the correct codeword. If the result of the inner decoder at position j is correct ($\hat{f}(P_j) = y_j$) then $\hat{Z}$ – the lower bound on the number of zeroes of $Q(x, \hat{f}(x))$ – is raised more than the increase in weighted degree, so decoding is more likely to be successful (it is possible to find situations where the increase in weighted degree is greater than the amount of possible zeroes, however, that will not be important in this paper). On the other hand, if the value is wrong then the weighted degree of Q is raised but the number of zeroes is constant which means that decoding is less likely to be successful. This means that the weight s_j should be chosen corresponding to how confident we are that the symbol at the j^{th} position is correct. One could say that s_j is the reliability of the value of y_j, and even though these reliabilities are not directly connected to the reliabilities in GMD decoding it turns out that the choice of reliabilities in Eqn. 7.3 is optimal also here. Only, since s_j is required to be an integer, we multiply by d and set

$$s_j := d - 2w_j. \tag{7.11}$$

The following theorem says that the error correcting capability of this algorithm meets the upper bound in Theorem 7.14.1. This also implies that Algorithm 7.9 solves the special generalized minimum distance decoding problem involved in the decoding of concatenated codes.

Theorem 7.16
Algorithm 7.15 with multiplicities given by Eqn. 7.11 successfully recovers the correct codeword if

$$\hat{B} < dD/2$$

where $\hat{B}$ is the badness of the error pattern as defined in Def. 7.3 and dD is the designed minimum distance of the concatenated code. □

Proof: See Appendix 7.10.1.

The theorem implies that when d is even or when erasures can occur the choice of multiplicities in Eqn. 7.11 is optimal in the sense that it maximizes the weight of the smallest non-correctable error pattern.

The fact that GMD decoding requires several decoding attempts to decode a concatenated code up to half the designed distance makes the result that this can be done using only a single run of Algorithm 7.9 a little surprising. However, with respect to complexity (at the time of writing) we have that $C = \mathrm{O}(d^2 N)$ giving a total complexity of $\mathrm{O}(d^5 N^2)$ which is more than the complexity of the $(d+1)/2$ Reed-Solomon decodings used in GMD decoding.

When comparing GMD decoding with Algorithm 7.15 one could be lead to believe that all error patterns corrected by one of the methods are also corrected by the other. However, Example 7.22 at the end of the next section shows that this is not true.

7.7 The d odd case

In the previous section it was shown that Algorithm 7.15 with the multiplicities in Eqn. 7.11 corrects the greatest possible number of errors when using the inner decoder of Assumption 7.2 if d (the minimum distance of the inner code) is even or if erasures can occur.

So in this section it will be assumed that erasures cannot occur and that d is odd. In this case Theorem 7.14 leaves room for improvement, however, in Section 7.4 it was shown that GMD decoding generally cannot correct beyond half the product bound in the worst case. In this section we will find a method for calculating the error-correcting capability when using Algorithm 7.15 and an explicit expression for the asymptotic error correcting capability.

In this context asymptotic will mean that we have an infinite sequence of concatenated codes where the outer codes are Reed-Solomon codes with parameters $(N_1, K_1), (N_2, K_2), \ldots$ with $N_a \to \infty$ and $K_a/N_a \to \infty$ for $a \to \infty$ for some constant κ. The inner codes will be assumed to have fixed minimum distance d and dimension suitable for the alphabet size of the corresponding outer code. For each inner code we will have a bound on the list error-correcting capability, B_a^*, meaning that all error patterns of badness less than B_a^* will be corrected. The asymptotic fractional list

error-correcting capability is then β^* such that $B_a^*/N_a \to \beta^*$ for $a \to \infty$. Notice that β^* is only fractional with respect to the length of the outer code, not with respect to the total code length. For example, this means that a fractional error-correcting capability of half the designed distance is $d(1-\kappa)/2$.

We have the following upper bound which is better than the bound in Theorem 7.14 for large information rates (large κ).

Theorem 7.17
The asymptotic list error-correcting capability of Algorithm 7.15 is at most

$$d(1-\sqrt{\kappa})$$

where d is the minimum distance of the inner code and κ is the information rate of the outer code. □

Proof:
Suppose that errors occur such that d errors occur in a number, T_a, of inner word such that these are changed into wrong inner codewords and suppose that $T_a/N_a \to \gamma$ for $a \to \gamma$ for some constant γ. Since each inner word is a codeword, they must all have the same reliability and be assigned the same multiplicity, σ_a. By Theorem 7.13.2 the outer codewords cannot be recovered by the outer decoder for γ exceeding $1-\sqrt{\kappa}$. So the overall fractional error correcting capability is at most $d(1-\sqrt{\kappa})$. ■

In this section we will prove an asymptotic bound on the list error correcting capability of Algorithm 7.15. In order to do this it will be necessary to multiply the multiplicities of Eqn. 7.11 by some integer, σ, which will be an additional parameter of the algorithm. Thus the multiplicities are chosen to be

$$s_j := \sigma(d - 2w_j). \tag{7.12}$$

The following algorithm determines whether all error patterns with at most a given badness can be list decoded.

Algorithm 7.18
Let a code be the concatenation of an inner code of minimum distance d and an outer code which is a (N, K) Reed-Solomon code.

Input: $\sigma, B \in \mathbb{N}$.

Output: "true" only if all error patterns of badness at most B will be corrected by Algorithm 7.15 with multiplicities given by Eqn. 7.12.

for $Z_\sigma = dN - 2B, \ldots, dN - B$
  $C(B, Z_\sigma) \leftarrow \sigma(\sigma(d+1)+1)(B + Z_\sigma) - \frac{\sigma d(\sigma(d+2)+1)}{2} N$
  if $\ell(B, Z_\sigma) \geq \sigma Z_\sigma$ **then return** *"false"*
end
return *"true"*

where $\ell(B, Z_\sigma)$ is the value of ℓ given by Eqn. 7.8 with $C = C(B, Z_\sigma)$. □

The correctness of this algorithm is proved in Appendix 7.10.2. A corollary of the algorithm is a non trivial bound on the maximal number of codewords whose difference to a given word has at most a specified badness:

Corollary 7.19
If Algorithm 7.18 returns "true" for a given badness, B, then the number of codewords in the output of Algorithm 7.15 with multiplicities given by Eqn. 7.12 is at most

$$\left\lfloor \frac{\sqrt{1 + 8C_{\max}(B)/(K-1)} - 1}{2} \right\rfloor$$

where

$$C_{\max}(B) := \max\{C(B, Z_\sigma) \mid Z_\sigma \in \{dN - 2B, \ldots, dN - B\}\}. \qquad (7.13)$$

□

Proof:
By the proof of Algorithm 7.18, the maximal value of C in Algorithm 7.9 for a received word of badness B and with $\hat{Z}/\sigma = Z_\sigma$ is $C(B, Z_\sigma)$. The possible values of $\hat{Z}/\sigma$ are $\{dN - 2B, \ldots, dN - B\}$ so $C_{\max}(B)$ is the overall maximal value of C for badness B. By Theorem 7.10 the maximal number of codewords in the output of algorithm 7.9 (and thereby also of Algorithm 7.15) is $\lambda - 1$ satisfying (by Eqn. 7.9)

$$\lambda(\lambda - 1)(K - 1) \leq C_{\max}(B)$$

which gives the corollary. ■

Furthermore, the following theorem gives a lower bound on the asymptotic fractional list error-correcting capability:

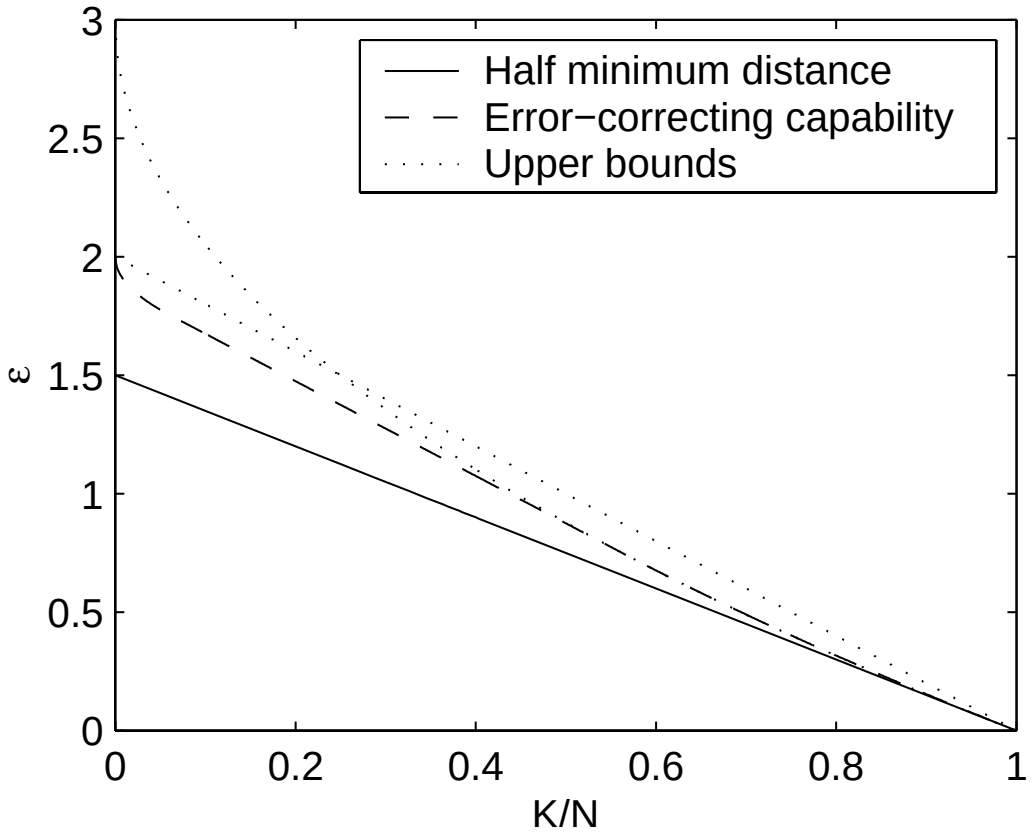

Figure 7.1: *Asymptotic fractional list error-correcting capability as explained in the text for $d = 3$ and multiplicities in 7.12. The upper bounds in Theorem 7.14 and 7.17 are shown as dotted lines.*

Theorem 7.20
Suppose that Algorithm 7.15 with multiplicities given by Eqn. 7.12 is used for list decoding a sequence of concatenated codes and that $\sigma = \sigma_a$ is chosen such that $\sigma_a/N_a^\gamma \to c$ for some (small) constant $\gamma > 0$ and a constant $c > 0$. Error patterns with a fractional badness β are corrected for $\beta < \beta^*$ where

$$\beta^* = \begin{cases} \frac{d+1}{2} - \sqrt{\kappa} & \text{for } 0 \le \kappa < \frac{1}{(d+1)^2} \\ \frac{d^2+2d}{2(d+1)} - \frac{d+1}{2}\kappa & \text{for } \frac{1}{(d+1)^2} \le \kappa \le \frac{d^2}{(d+1)^2} \\ d - d\sqrt{\kappa} & \text{for } \frac{d^2}{(d+1)^2} < \kappa \le 1 \end{cases}$$

□

Proof: See Appendix 7.10.3.

Figure 7.1 shows this error-correcting capability in the case where $d = 3$ together with the upper bounds derived in Theorem 7.14 and 7.17.

Z	B_0^*	B_1^*	B_3^*	C	ℓ	$\lambda - 1$
24	4	12	0	36	20	2
25	14/3	11	1/3	41	21	2
26	16/3	10	2/3	46	23	2
27	6	9	1	51	24	3
28	20/3	8	4/3	56	26	3
29	22/3	7	5/3	61	27	3
30	8	6	2	66	28	3
31	26/3	5	7/3	71	29	3
32	28/3	4	8/3	76	31	3
33	10	3	3	81	32	4
34	32/3	2	10/3	86	33	4
35	34/3	1	11/3	91	34	4
36	12	0	4	96	35	4

Table 7.1: The table shows the intermediate results when using Algorithm 7.18 in Example 7.21 for badness $B = 12$. For each valid value of Z, the table gives the optimal assignment of Eqn. 7.25 as well as the resulting value of $C := C(12, Z)$ and the corresponding $\ell := \ell(12, Z)$. Also $\lambda - 1$ (as calculated by Eqn. 7.9) is given which provides an upper bound on the number of codewords in the output of the decoding algorithm (Algorithm 7.15).

Notice that Theorem 7.20 gives a qualitative description of the result of Algorithm 7.18 and while the theorem is useful in that it gives an explicit expression it is the algorithm which is useful in practice when calculating a good lower bound on the list error-correcting capability of Algorithm 7.15 for a given code.

The following gives an example where Algorithm 7.18 is used to determine a lower bound on the list error correcting capability of Algorithm 7.15:

Example 7.21
Let the inner code have minimum distance $d = 3$ and let the outer code be a $(16, 9)$ Reed-Solomon code. The minimum distance of the concatenated code is then at least 24. In Algorithm 7.18, let $\sigma = 1$ and $Z := Z_\sigma$. For badness $B = 12$, the value of Z is an integer with $24 \leq Z \leq 36$. Table 7.1 shows the intermediate results in Algorithm 7.18. As seen by the table, $Z > \ell$ in all cases, so all error patterns of badness up to 12 are corrected. On the other hand, we have $C(13, 31) = 76$ (for $B_0^* = 9$, $B_1^* = 4$, $B_3^* = 3$). This gives $\ell(13, 31) = 31$ so such an error pattern with badness 13 is not guaranteed to be corrected.

K	$\lfloor \frac{dD-1}{2} \rfloor$	B	$\lambda - 1$
2	22	26	13
3	20	24	9
4	19	22	7
5	17	20	6
6	16	18	5
7	14	16	5
8	13	14	4

K	$\lfloor \frac{dD-1}{2} \rfloor$	B	$\lambda - 1$
9	11	12	4
10	10	10	4
11	8	8	3
12	7	7	3
13	5	5	3
14	4	4	3
15	2	2	3

Table 7.2: For each dimension, K, in Example 7.21 the error-correcting capability of GMD decoding (half the designed minimum distance) is shown together with the list error-correcting capability, B, of Algorithm 7.15 and an upper bound on the number of codewords in the output. In both cases the capability is for error patterns of badness up to at most the given limit. For all dimensions, $C_{\max}(B) = 96$.

If we let the dimension, K, of the outer code vary between 2 and 15, the list error-correcting capabilities of Table 7.2 are found. As seen by the table, error patterns of badness up to half the designed distance are always corrected and for the low dimensions the code is decoded also beyond half the designed distance giving an improvement over GMD decoding. □

The following example shows how Algorithm 7.15 can be used for decoding a given received word with a given concatenated code.

Example 7.22
Let $\mathcal{C}_1$ be the binary $(7, 4, 3)$ Hamming code with generator matrix

$$G := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Let $\mathbb{F}_{16}$ be constructed with primitive element α satisfying $\alpha^4 = \alpha + 1$ and let $\mathcal{C}_2$ be the $(16, 9)$ Reed-Solomon code over $\mathbb{F}_{16}$ constructed by evaluating

$$P_1, \ldots, P_{16} = 0, 1, \alpha, \ldots, \alpha^{14}.$$

In constructing the concatenated code, let the inner encoding be done by the mapping, $\mathrm{enc}_{\mathcal{C}_1} : \mathbb{F}_{16} \to \mathcal{C}_1$ where

$$\mathrm{enc}_{\mathcal{C}_1}(a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3) = (a_0, a_1, a_2, a_3)G.$$

j	r_j	v_j	y_j	s_j
1	1000000	0000000	0	1
2	0000000	0000000	0	3
3	1011001	0011001	α^6	1
4	0000001	0000000	0	1
5	0000000	0000000	0	3
6	0001000	0000000	0	1
7	0000000	0000000	0	3
8	0010100	0010110	α^2	1

j	r_j	v_j	y_j	s_j
9	0000000	0000000	0	3
10	0100000	0000000	0	1
11	0000000	0000000	0	3
12	0000000	0000000	0	3
13	0111010	0101010	α^9	1
14	0001100	1001100	α^{14}	1
15	0000000	0000000	0	3
16	0000000	0000000	0	3

Table 7.3: First received word of Example 7.22. For each outer position, j, the received inner word, r_j, is shown together with the output, v_j, of the inner decoder and the corresponding outer symbol, $y_j = \mathrm{enc}_{\mathcal{C}_1}^{-1}(v_j)$, and multiplicity, $s_j := d - \mathrm{d}(r_j, v_j)$, for the outer decoder.

The concatenated code is then a binary $(128, 27, 24)$ *code (for example,* $x(x+1)(x^2+x+1)(x^4+x+1)$ *gives a binary outer word which is encoded into a codeword of weight* 24*).*

Suppose that the zero word is sent and that an error pattern of badness 12 (and weight 16) occurs such that the inner words in the second column of Table 7.3 are received. For each inner word, the table describes the processing of the inner decoder. Now, Algorithm 7.9 is applied on the input points (P_j, y_j) *with multiplicities* s_j *for* $j = 1, \ldots, 16$*. The following is found to be a valid* Q*-polynomial:*

$$\begin{aligned} Q(x,y) =& (\alpha^{13} + \alpha^{14}x + \alpha^4 x^4 + \alpha^5 x^5 + \alpha^{12}x^6 + \alpha^{13}x^7 + \alpha^7 x^8 + \alpha^8 x^9 + \\ & \alpha^{13}x^{10} + \alpha^{14}x^{11} + \alpha x^{12} + \alpha^2 x^{13} + \alpha^7 x^{14} + \alpha^8 x^{15} + \alpha^{11}x^{16} + \\ & \alpha^{12}x^{17})y + (\alpha^{12} + \alpha^2 x + x^2 + \alpha^8 x^3 + \alpha^5 x^5 + \alpha^3 x^6 + \alpha^2 x^7 + \\ & \alpha^{10}x^8 + \alpha x^9)y^2 + \alpha y^3 \end{aligned}$$

It is seen directly that y *is a factor, so the zero word is in the output. There is a second factor of the form* $y - f(x)$*, but it corresponds to a codeword at distance* 30 *from the received word.*

Notice that in general the GMD algorithm fails to recover the correct codeword since decoding will be attempted on the following words (reliabilities are $s_j/3$*):*

$$\begin{array}{cccccccccccccccc} 0 & 0 & \alpha^6 & 0 & 0 & 0 & 0 & \alpha^2 & 0 & 0 & 0 & 0 & \alpha^9 & \alpha^{14} & 0 & 0 \\ ? & 0 & ? & ? & 0 & ? & 0 & ? & 0 & ? & 0 & 0 & ? & ? & 0 & 0 \end{array}$$

Suppose that the zero word is sent again and that 13 errors occur such that the inner words in the second column of Table 7.4 are received.

j	r_j	v_j	y_j	s_j
1	0000000	0000000	0	3
2	0000000	0000000	0	3
3	0100000	0000000	0	1
4	0000000	0000000	0	3
5	0010110	0010110	α^2	3
6	0000000	0000000	0	3
7	0000000	0000000	0	3
8	0100000	0000000	0	1

j	r_j	v_j	y_j	s_j
9	0000000	0000000	0	3
10	1110000	1110000	α^{10}	3
11	0000010	0000000	0	1
12	0000000	0000000	0	3
13	0000000	0000000	0	3
14	0001000	0000000	0	1
15	0000000	0000000	0	3
16	0101010	0101010	α^9	3

Table 7.4: Second received word of Example 7.22. For each outer position, j, the received inner word, r_j, is shown together with the output, v_j, of the inner decoder, the corresponding outer symbol, y_j, and multiplicity, s_j.

Algorithm 7.9 is applied on the input points (P_j, y_j) with multiplicities s_j for $j = 1, \ldots, 16$. The following Q-polynomial is found:

$$\begin{aligned} Q(x,y) =& (\alpha^4 x^3 + \alpha^9 x^4 + \alpha^2 x^5 + \alpha^4 x^6 + \alpha^{11} x^7 + \alpha^{10} x^8 + \alpha^6 x^9 + \alpha^3 x^{10} + \\ & \alpha^4 x^{11} + \alpha^4 x^{12} + \alpha^2 x^{14} + \alpha^{13} x^{15} + \alpha x^{16} + \alpha^3 x^{17} + \alpha^7 x^{18} + \alpha x^{19} + \\ & \alpha^4 x^{20} + \alpha^8 x^{22} + \alpha^{10} x^{23} + \alpha^{12} x^{24} + \alpha^{13} x^{25} + \alpha x^{27} + \alpha^8 x^{28} + \\ & \alpha^3 x^{29} + \alpha^2 x^{30} + x^{31}) + (\alpha^{14} x^2 + \alpha^{11} x^3 + \alpha^9 x^4 + \alpha^7 x^5 + \alpha^9 x^6 + \\ & x^7 + \alpha^4 x^8 + \alpha^9 x^9 + \alpha^2 x^{10} + \alpha^7 x^{12} + \alpha^5 x^{13} + \alpha^{10} x^{14} + \alpha^{11} x^{15} + \\ & \alpha^4 x^{16} + \alpha^7 x^{17} + x^{18} + \alpha^{10} x^{19} + \alpha^{13} x^{20} + \alpha^9 x^{21} + \alpha^{13} x^{22}) y + \\ & (\alpha^7 x + \alpha^9 x^2 + \alpha^2 x^3 + \alpha^6 x^5 + \alpha x^6 + \alpha^{12} x^7 + \alpha^{13} x^8 + \alpha x^9 + \\ & \alpha^{14} x^{10} + \alpha^{13} x^{11} + \alpha^2 x^{12} + \alpha^{13} x^{13} + \alpha^5 x^{14}) y^2 + (\alpha^7 + \alpha^{14} x + \\ & \alpha^4 x^2 + \alpha^{12} x^3 + \alpha^3 x^4 + \alpha^{14} x^5 + \alpha^8 x^6) y^3. \end{aligned}$$

This polynomial has no factors in the form $y - f(x)$ so the decoding fails to recover the correct codeword.

On the other hand, GMD decoding attempts decoding the following words

$$\begin{array}{cccccccccccccccc} 0 & 0 & 0 & 0 & \alpha^2 & 0 & 0 & 0 & 0 & \alpha^{10} & 0 & 0 & 0 & 0 & 0 & \alpha^9 \\ 0 & 0 & ? & 0 & \alpha^2 & 0 & 0 & ? & 0 & \alpha^{10} & ? & 0 & 0 & ? & 0 & \alpha^9 \end{array}$$

So the correct codeword is successfully recovered by the first attempt.

It is worth noticing that an algorithm attempting both GMD decoding and Algorithm 7.15 and then taking the candidate closest to the received word as the result would have succeeded in correcting both error patterns of this example. □

7.8 Improvement for special cases

In the previous sections in has been seen that Assumption 7.2 limits the error-correction capability of Algorithm 7.9 when applied to concatenated codes. In this section we will, therefore, drop Assumption 7.2 and allow the inner decoder to be a list decoder.

The analysis in this section can be done also in the presence of erasures, but it turns out that erasures will behave different than in the previous sections. In the previous sections the weight of an erasure was $1/2$ since that was the value that was natural to use in the error-correction bounds. In this section the most straightforward value to use for the weight of erasures would be $(q-1)/q$ (to have an analogue to Lemma 7.34). However, in order to make the analysis clearer it will be assumed in the rest of this section that erasures do not occur.

Furthermore, it turns out that the concept of badness does not generalize in a nice way. Therefore, the results in this section will consider only the weight of the smallest error pattern which cannot be corrected, which means that only an analogue to worst case error patterns will be considered.

In this section it will be assumed that the inner decoder list decodes the inner code up to some specified radius:

Assumption 7.23
Let $\mathcal{C}_1 \subseteq \mathbb{F}_q^n$ be the inner code in a concatenated code. For some given fixed rational number R with $0 \leq R \leq n$ the inner decoder outputs the following set for any received word, $r_j \in \mathbb{F}_q^n$

$$\mathcal{V}_j := \{v \in \mathcal{C}_1 \mid \mathrm{d}(v, r_j) < R\}.$$

□

Notice that if $R = d/2$ then Assumption 7.23 essentially reduces to Assumption 7.2. Furthermore, it should be noted that if we only care about the output of the decoder, we could require R to be an integer. However, allowing R to take on arbitrary rational values will be used in the outer decoding to adjust the weighting of the outer symbol corresponding to each inner codeword in the output.

If $R > d/2$ then $\mathcal{V}_j$ may contain several codewords. Enumerate the elements of $\mathcal{V}_j$ as $v_{j,1}, \ldots, v_{j,u(j)}$ where $u(j) = |\mathcal{V}_j|$. Each of these corresponds to an element of $\mathbb{F}_{q^k}$. Let the set of $\mathbb{F}_{q^k}$-elements corresponding to

the codewords in $\mathcal{V}_j$ be denoted by $\mathcal{Y}_j$:

$$\mathcal{Y}_j := \{y_{j,1}, \ldots, y_{j,u(j)}\}$$

where $y_{j,i} = \mathrm{enc}_{\mathcal{C}_1}^{-1}(v_{j,i})$ for $i = 1, \ldots, u(j)$. Each of these elements has a reliability determined by the distance $\mathrm{d}_{\mathcal{A}}(r_j, v_{j,i})$ and we want to associate a multiplicity, $s_{j,i}$ with each $y_{j,i}$ for use in Algorithm 7.9. A natural generalization of Eqn. 7.11 would be to let $s_{j,i}$ equal $2R - 2\mathrm{d}(r_j, v_{j,i})$, however, to allow R to be any positive rational number, let

$$s_{j,i} := \sigma(R - \mathrm{d}(r_j, v_{j,i})) \tag{7.14}$$

where σ is an integer parameter which satisfies that the denominator of R when written as an irreducible fraction divides σ.

The following algorithm (which generalizes Algorithm 7.15) shows how to combine an inner decoder as specified in Assumption 7.23 with the list decoding method for Reed-Solomon codes in Algorithm 7.9 into a list decoding method for concatenated codes.

Algorithm 7.24

1. *Use an inner decoder as specified in Assumption 7.23 on each component of the received word, r, such that outer symbols, $y_{j,1}, \ldots, y_{j,u(j)} \in \mathbb{F}_{q^k}$, are obtained for each outer position, $j \in \{1, \ldots, N\}$.*

2. *Use Algorithm 7.9 with the following input: Let $m := \sum_{j=1}^{N} u(j)$ and let the interpolation points be $(P_j, y_{j,i})$ for $j \in \{1, \ldots, N\}$ and $i \in \{1, \ldots, u(j)\}$. The corresponding multiplicities $s_{j,i}$ are given by Eqn. 7.14.*

□

In the following we will state a method to calculate a good lower bound on the error-correcting capability of the above algorithm. It turns out that this is generally hard since it requires (partial) knowledge of the distribution of distances from a given word, $r \in \mathbb{F}_q^n$, to the codewords of the inner code (this equals the weight distribution of the coset of the inner code which contains $-r$). The following notation will be used:

$$U_i(\mathcal{C}_1; r) := |\{c \in \mathcal{C}_1 \mid \mathrm{d}(r, c) = i\}|$$

where $\mathcal{C}_1 \subset \mathbb{F}_q^n$, $r \in \mathbb{F}_q^n$, and $i \in \{0, \ldots, n\}$.

Consider a received inner word, r_j, in step 1 of Algorithm 7.24. Each candidate which the inner decoder outputs will contribute to the number of linear constraints, C, in the list decoding of the outer code. The total contribution corresponding to r_j depends on $U_i(\mathcal{C}_1; r_j)$ for $i = 0, \ldots, \lfloor R \rfloor$ such that the contribution is

$$\sum_{i=0}^{\lfloor R \rfloor} \binom{\sigma(R-i)+1}{2} U_i(\mathcal{C}_1; r_j).$$

For a given error weight, w, of an inner word and a given decoding radius, R, let $C_{\max}(w, R)$ denote the maximal contribution to C. That is

$$C_{\max}(w, R) := \max \left\{ \sum_{i=0}^{\lfloor R \rfloor} \binom{\sigma(R-i)+1}{2} U_i(\mathcal{C}_1; r) \mid r \in \mathcal{S}(\mathcal{C}_1, w) \right\} \tag{7.15}$$

where

$$\mathcal{S}(\mathcal{C}_1, w) := \{v \in \mathbb{F}_q^n \mid \exists c \in \mathcal{C}_1 : \mathrm{d}(v, c) = w\} \tag{7.16}$$

is the set of words at distance w from some codeword in $\mathcal{C}_1$.

Before continuing, it may be helpful with an example which relates the concepts defined above to the special case considered in the previous sections:

Example 7.25

Suppose that $R = d/2$. If $\sigma = 2$ then this is the situation that has been analyzed in the previous sections, however, for this example let σ be any positive even integer.

Let r_j be a received inner word, and let $\hat{c}_j$ be the corresponding sent word such that $\hat{w}_j := \mathrm{d}(r_j, \hat{c}_j)$ errors occurred. The reason why the analysis was possible in the previous sections is that we know $U_i(\mathcal{C}_1; r_j)$ for all necessary cases.

By the minimum distance of the inner code, $\mathcal{C}_1$, we have that $U_i(\mathcal{C}_1; r_j)$ is non-zero for at most one $i < d/2$. If $\hat{w}_j < d/2$ then $U_i(\mathcal{C}_1, r_j) = 1$ for $i = \hat{w}_j$. If $\hat{w}_j = d/2$ then $U_i(\mathcal{C}_1, r_j) = 0$ for all $i < d/2$. If $d/2 < \hat{w}_j \leq d$ then $U_i(\mathcal{C}_1, r_j) = 0$ for all $i < d - \hat{w}_j$. This implies that

$$C_{\max}(d/2 - i, d/2) = C_{\max}(i, d/2) = \binom{\sigma(d/2-i)+1}{2}$$

for $i \in \{0, \ldots, \lfloor d/2 \rfloor\}$.

Furthermore, it is seen that

$$C_{\max}(i, d/2) \leq C_{\max}(d, d/2)$$

for $i \in \{d, \ldots, n\}$. *This and the fact that the contribution to* $\hat{Z}$ *is 0 for all* $\hat{w}_j \geq d$ *means that in a worst case situation it can be assumed that* $\hat{W}_i = 0$ *for* $i > d$. *Therefore,*

$$C \leq \sum_{i=0}^{\lfloor d/2 \rfloor} (\hat{W}_i + \hat{W}_{d-i}) \binom{\sigma(d/2 - i) + 1}{2}.$$

The right side is the expression found in Eqn. 7.23 for a worst case error pattern (where $\hat{W}_i = \hat{B}_i$ *for* $i = 0, \ldots, d$ *and* $\hat{W}_i = 0$ *for* $i > d$*).* □

The following algorithm determines whether all error patterns with at most a given weight can be list decoded.

Algorithm 7.26
Let a code be the concatenation of an inner code, $\mathcal{C}_1 \subseteq \mathbb{F}_q^n$ *and an outer code which is a* (N, K) *Reed-Solomon code.*
Input: $\sigma, W \in \mathbb{N}$. *Inner decoding radius,* R.
Output: "true" only if all error patterns of weight at most W *will be corrected by Algorithm 7.24.*

1. *Calculate* $C_{\max}(w, R)$ *(Eqn. 7.15) for* $w = 0, \ldots, n$.

2. *For* $Z = \sigma(RN - W), \ldots, \lfloor \sigma(RN - WR/n) \rfloor$ *do the following:*

   - *Let* $C(W, Z, R)$ *be the solution of the following linear programming problem:*

$$C(W, Z, R) := \max \sum_{i=0}^{n} C_{\max}(i, R) W_i^*$$

where W_i^* for $i \in \{0, \ldots, n\}$ satisfy the constraints

$$\sum_{i=0}^{n} W_i^* = N$$

$$\sum_{i=0}^{n} iW_i^* = W$$

$$\sum_{i=0}^{\lfloor R \rfloor} (R-i)W_i^* = Z/\sigma$$

$$W_i^* \geq 0.$$

- *Let $\ell(W, Z, R)$ be the value of ℓ given by Eqn. 7.8 with $C = C(W, Z, R)$. If $\ell(W, Z, R) \geq Z$ then return "false" and stop.*

3. Return "true".

□

The correctness of this algorithm is proved in Appendix 7.10.4.

It should be mentioned that Algorithm 7.26 can be computationally expensive. The first step (calculating the $C_{\max}$'s) often requires calculating the weight distribution of all cosets of the inner code, and this limits the size of inner codes that can be treated (at the time of writing, binary inner codes of length beyond 32 would start to give problems). On the other hand, using Algorithm 7.26 is much more efficient than looking at possible error patterns for the entire concatenated code.

A side effect of the algorithm is a non-trivial bound on the maximal number of codewords whose difference to a given word has at most a specified distance:

Corollary 7.27
If Algorithm 7.26 returns "true" for a given error weight, W, then the number of codewords in the output of Algorithm 7.24 with inner decoding radius R is at most

$$\left\lfloor \frac{\sqrt{1 + 8C_{\max}(R)/(K-1)} - 1}{2} \right\rfloor$$

where

$$C_{\max}(R) := \max\{C(B, W, Z) \mid Z \in \{\sigma(RN - W), \ldots, \lfloor \sigma(RN - WR/n) \rfloor\}\}$$

□

Proof:
By the proof of Algorithm 7.26, the maximal value of C in Algorithm 7.9 for a received word with error weight W and with $\hat{Z} = Z$ is $C(W, R, Z)$. The possible values of $\hat{Z}$ are $\{\sigma(RN - W), \ldots, \lfloor \sigma(RN - WR/n) \rfloor\}$ so $C_{\max}(R)$ is the overall maximal value of C. By Theorem 7.10 the maximal number of codewords in the output of Algorithm 7.9 (and thereby also of Algorithm 7.24) is $\lambda - 1$ satisfying (by Eqn. 7.9)

$$\lambda(\lambda - 1)(K - 1) \leq C_{\max}(R)$$

which gives the corollary. ■

The following example illustrates the use of Algorithm 7.26 in determining a lower bound on the error-correcting capability of Algorithm 7.24:

Example 7.28
Let $\mathcal{C}_1$ be a $(9, 8, 2)$ binary parity check code and let $\mathcal{C}_2$ be a $(256, 65, 192)$ RS code over $\mathbb{F}_{2^8}$. The concatenated code has parameters $(2304, 520, \geq 384)$. Cosets of $\mathcal{C}_1$ have one of two weight distributions. If $r_j \in \mathcal{C}_1$ and $\bar{r}_j \in \mathbb{F}_2^9 \backslash \mathcal{C}_1$ then

i	0	1	2	3	4	5	6	7	8	9
$U_i(\mathcal{C}_1; r_j)$	1	0	36	0	126	0	84	0	9	0
$U_i(\mathcal{C}_1; \bar{r}_j)$	0	9	0	84	0	126	0	36	0	1

This shows that if the received inner word on some position is not a codeword then there are 9 inner codewords at distance 1 from the received inner word. We choose the decoding radius of the inner decoder to be $R := 8/7$ and the multiplicity multiplier to be $\sigma := 7$. The value of R is slightly greater than half the minimum distance of $\mathcal{C}_1$. The effect is that if the received inner word on some position is a codeword then the corresponding outer symbol is given multiplicity 8 in the outer decoding. This is specified by $8 \cdot 9/2 = 36$ linear equations. If the received inner word is not a codeword then each of the 9 outer symbols corresponding to an inner codeword at distance 1 are given multiplicity 1 — this is specified by 9 linear equations.

Since an even number of errors in an inner word gives a codeword while an odd number of errors give a non-codeword, the $C_{\max}(i, R)$'s are as follows:

Z	W_0^*	W_1^*	W_2^*	W_8^*	W_9^*	C	ℓ
683	61	195	0	0	0	3951	679
803	81	155	20	0	0	5031	771
1400	175	151/2	0	11/2	0	9216	1054
1874	937/4	0	0	3/4	21	8649	1020

Table 7.5: Optimal assignments of the W_i^*'s in Example 7.28 for some selected values of Z. The corresponding values of $C = C(W, Z, R)$ and $\ell := \ell(W, Z, R)$ are shown as well.

i	0	1	2	3	4	5	6	7	8	9
$C_{\max}(i, R)$	36	9	36	9	36	9	36	9	36	9

For a given value of W the following function should be maximized for each $Z \in \{\sigma(RN - W), \ldots, \sigma(RN - WR/n)\}$:

$$36(W_0^* + W_2^* + W_4^* + W_6^* + W_8^*) + 9(W_1^* + W_3^* + W_5^* + W_7^* + W_9^*)$$

where the W_i^*'s for $i \in \{0, \ldots, 9\}$ must be positive rational numbers satisfying the constraints

$$\begin{aligned} W_0^* + W_1^* + \ldots + W_9^* &= 256 \\ W_1^* + 2W_2^* + 3W_3^* + \ldots + 9W_9^* &= W \\ 8W_0^* + W_1^* &= Z \end{aligned}$$

For $W = 195$ we have that $Z \in \{683, \ldots, 1874\}$. In Table 7.5 the optimal assignment of W_i^*'s is given for some selected values of Z (the W_i^*'s which are not shown are all 0). It turns out that for all values of Z the value of $\ell = \ell(W, Z, R)$ is less than Z. So all error patterns of weight up to 195 can be corrected. The overall maximal value of $C(W, Z, R)$ is $C_{\max}(R) = 9216$ which gives a measure of the worst case running time of the decoding and Corollary 7.27 gives that at most 16 codewords can be in the output of Algorithm 7.24 in this case.

On the other hand, for $W = 196$ and $Z = 676$ the optimal assigment of W_i^*'s is $W_0^* = 60$ and $W_1^* = 196$ which gives $C = 3924$ and $\ell = 676 = Z$ so such an error pattern is not guaranteed to be corrected.

In Theorem 7.16 it was shown that error patterns up to weight 191 can be corrected by choosing $R = 1$ and $\sigma = 2$. The overall maximal value of C is 768 in this case (actually, in the error-only case we can choose $R = 1$ and $\sigma = 1$ and correct 191 errors with a maximal value of C of 256). So

the error-correcting capability of the choice above is better at the cost of slower decoding.

The same improvement cannot be obtained simply by choosing $R = 1$ *and* $\sigma = 8$*. In that case the maximal value of* C *is* 9216*, but the error-correcting capability remains at 191.* □

If the inner code is fixed it does not make sense to obtain results which are asymptotic with respect to the code length, however, results which are asymptotic with respect to σ can be obtained. As usual for asymptotic results there may be a considerable difference between the asymptotic behaviour and the behaviour for a given fixed value. — The purpose of calculating an asymptotic error-correcting capability with respect to σ is that this in practice serves an upper bound of the error-correcting capability of Algorithm 7.24.

Notice that for $\sigma \to \infty$ we have $C_{\max}(w, R)/\sigma^2 \to C_{\max,\infty}(w, R)$ with

$$C_{\max,\infty}(w, R) := \max \left\{ \sum_{i=0}^{\lfloor R \rfloor} \frac{(R-i)^2}{2} U_i(\mathcal{C}_1; r) \mid r \in \mathcal{S}(\mathcal{C}_1, w) \right\} \qquad (7.17)$$

where $\mathcal{S}(\mathcal{C}_1, w)$ is the set of words at distance w from some codeword in $\mathcal{C}_1$ (see Eqn. 7.16).

We then have the following algorithm:

Algorithm 7.29
Let a code be the concatenation of an inner code, $\mathcal{C}_1 \subseteq \mathbb{F}_q^n$ *and an outer code which is a* (N, K) *Reed-Solomon code.*
Input: $W \in \mathbb{N}$*. Inner decoding radius,* R*.*
Output: "true" only if all error patterns of weight at most W *will be corrected by Algorithm 7.24 for* $\sigma \to \infty$*.*

1. *Calculate* $C_{\max,\infty}(w, R)$ *(Eqn. 7.17) for* $w = 0, \ldots, n$*.*
2. *For all* $Z_\infty \in \{b(RN - W), \ldots, \lfloor b(RN - WR/n) \rfloor\}/b$ *where* $R = a/b$ *for* $a, b \in \mathbb{N}$ *and* $\gcd(a, b) = 1$*, do the following:*
   - *Let* $C_\infty(W, Z_\infty, R)$ *be the solution of the following linear programming problem:*

$$C_\infty(W, Z_\infty, R) := \max \sum_{i=0}^{n} C_{\max,\infty}(i, R) W_i^*$$

where W_i^* *for* $i \in \{0, \ldots, n\}$ *satisfy the constraints*

$$\begin{aligned} \sum_{i=0}^{n} W_i^* &= N \\ \sum_{i=0}^{n} i W_i^* &= W \\ \sum_{i=0}^{\lfloor R \rfloor} (R-i) W_i^* &= Z_\infty \\ W_i^* &\geq 0. \end{aligned}$$

- *Let* $\ell_\infty(W, Z_\infty, R) := \sqrt{2(K-1)C(W, Z_\infty, R)}$. *If* $\ell_\infty(W, Z_\infty, R) \geq Z_\infty$ *then return "false" and stop.*

3. *Return "true".*

□

The correctness of this algorithm is seen from the correctness of Algorithm 7.26 when noticing that Lemma 7.11.2 implies that ℓ as defined in Eqn. 7.8 satisfies $\ell/\sigma \to \sqrt{2(K-1)C_\infty}$ if $C/\sigma^2 \to C_\infty$ for $\sigma \to \infty$.

Example 7.30
In this example 4 different binary concatenated codes will be compared. All the codes have the same dimension and for all of them the outer code is a Reed-Solomon code over $\mathbb{F}_{2^8}$ *where the length is adjusted to give similar lengths of the concatenated codes. The inner codes are (1) the (9,8,2) even weight code, (2) a (12,8,3) shortened Hamming code, (3) a (13,8,4) extended shortened Hamming code, and (4) a (16,8,5) code. All these codes have the smallest possible length for a linear binary code of the given dimension and minimum distance. Schematically, the parameters of the codes are*

1. $(9, 8, 2) * (256, 75, 182) \to (2304, 600, \geq 364)$
2. $(12, 8, 3) * (192, 75, 118) \to (2304, 600, \geq 354)$
3. $(13, 8, 4) * (177, 75, 103) \to (2301, 600, \geq 412)$
4. $(16, 8, 5) * (144, 75, 70) \to (2304, 600, \geq 350)$

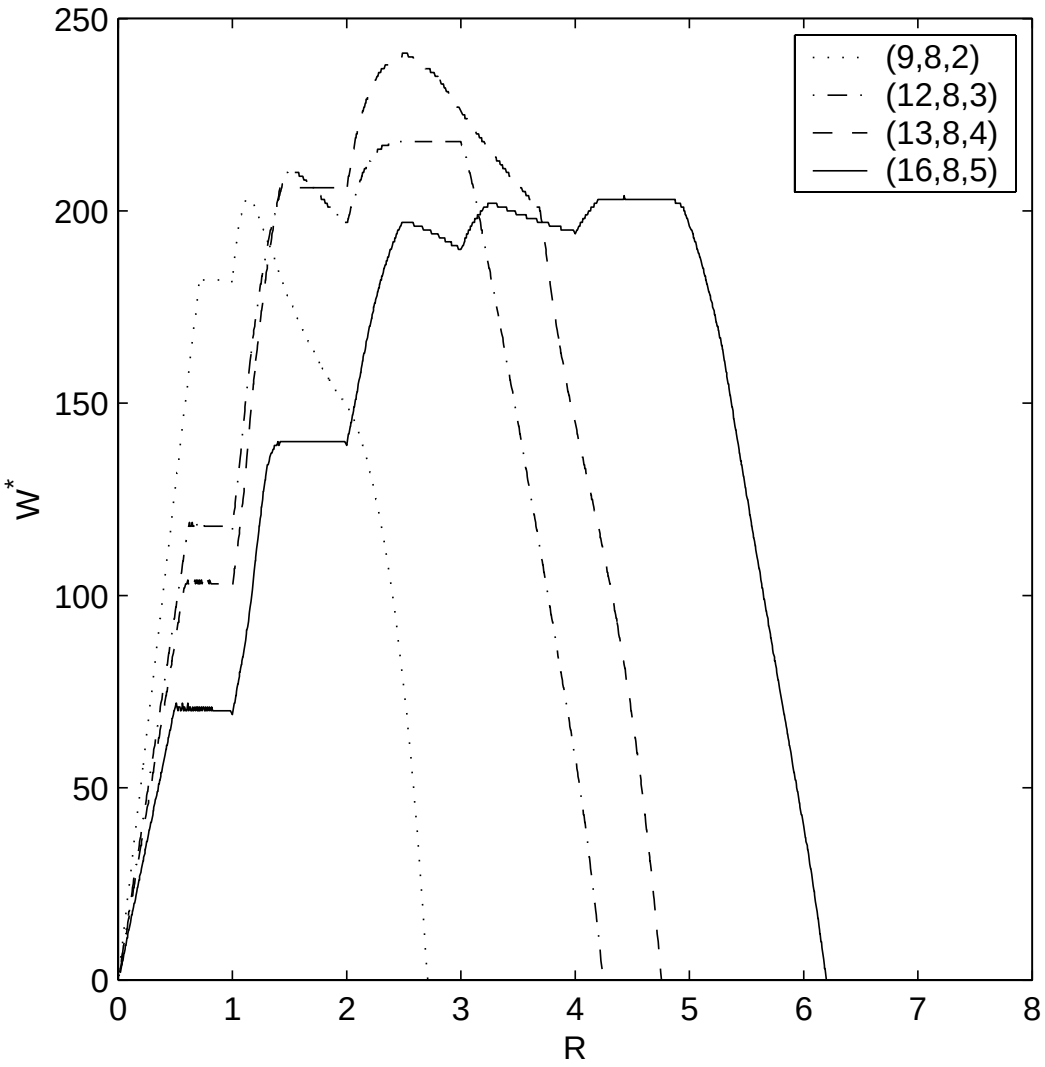

Figure 7.2: *Lower bounds on the asymptotic error-correcting capability of Algorithm 7.24 applied to the 4 binary concatenated codes of Example 7.30 for varying decoding radius of the inner codes.*

Figure 7.2 shows the asymptotic (with respect to σ) list error-correcting capability for $R \in [0; 7]$ as calculated by Algorithm 7.29.

For each of the codes, the figure gives space for a list error-correcting capability of Algorithm 7.24 which is well above half the minimum distance. However, the list error-correcting capability is related to the minimum distance such that codes with large minimum distance gives a large list error-correcting capability and vice versa.

From the figure it is also seen that the dependence between the list error-correcting capability and the inner decoding radius is so complicated that an explicit description is probably out of reach in general.

In order to compare the actual list error-correcting capability to the asymptotic limit, table 7.6 shows the list error-correcting capability obtained with the different codes for some selected values of R.

The table follows the general picture obtained from the asymptotic list error-correcting capabilities. However, the error-correcting capabilities are smaller since we are now looking at given (small) values of σ. For code 1 it seems that a slightly smaller radius than the one read from the asymptotic picture is to be preferred — other experiments indicates that this is a common behaviour. Code 3 is the one performing best. The cost penalty in the case where 230 errors are corrected is very significant with the implementation methods available at the moment of writing, but the performance of correcting a percentage of 12% errors beyond half the minimum distance is also a significant gain. □

The next example discusses the results of [14] where the inner code is a Hadamard code:

Example 7.31
Let m be a given positive integer and let a concatenated code be constructed by an inner Hadamard code[3] with parameters (q^m, m, d) over $\mathbb{F}_q$ and an outer Reed-Solomon code with parameters (q^m, K, D) over $\mathbb{F}_{q^m}$. We have $d = q^{m-1}$ and $D = q^m - K + 1$. Theorem 7 of [14] says that all error patterns of weight at most W can be corrected by Algorithm 7.24 with $R = q^{m-1}$

[3]The Hadamard codes of [14] are usual Hadamard codes with a 0-position added, so the code can safely be punctured in that position. Actually, an even better improvement can be made since Theorem 7 of [14] also can be proved with a first order generalized Reed-Muller code as inner code (parameters: $(q^m, m+1, q^{m-1})$).

Code	R	σ	W	$C_{\max}(R)$	$\lambda - 1$
1	1	1	181	256	2
1	4/3	3	154	2560	7
1	7/6	6	181	7168	13
1	8/7	7	184	9216	15
1	9/8	8	187	11520	17
1	9/8	∞	203	—	—
2	3/2	2	197	1152	5
2	3/2	4	203	4032	9
2	3/2	6	206	8640	14
2	3/2	∞	211	—	—
2	5/2	2	194	2088	7
2	5/2	∞	218	—	—
3	2	1	205	531	3
3	5/2	2	209	2655	7
3	5/2	4	224	9735	15
3	5/2	6	230	21240	23
3	5/2	∞	241	—	—
4	3	1	177	864	4
4	3	3	185	6480	12
4	3	∞	190	—	—

Table 7.6: List error-correcting capability of Algorithm 7.24 with the concatenated codes of Example 7.30 and selected values of the inner decoding radius, R. In each case, W, is the lower bound on the list error-correcting capability found by Algorithm 7.26 and $\lambda - 1$ is an upper bound on the number of codewords in the output. The worst case cost of the decoding is indicated by $C_{\max}(R)$ which is the maximal number of linear constraints involved in Algorithm 7.9.

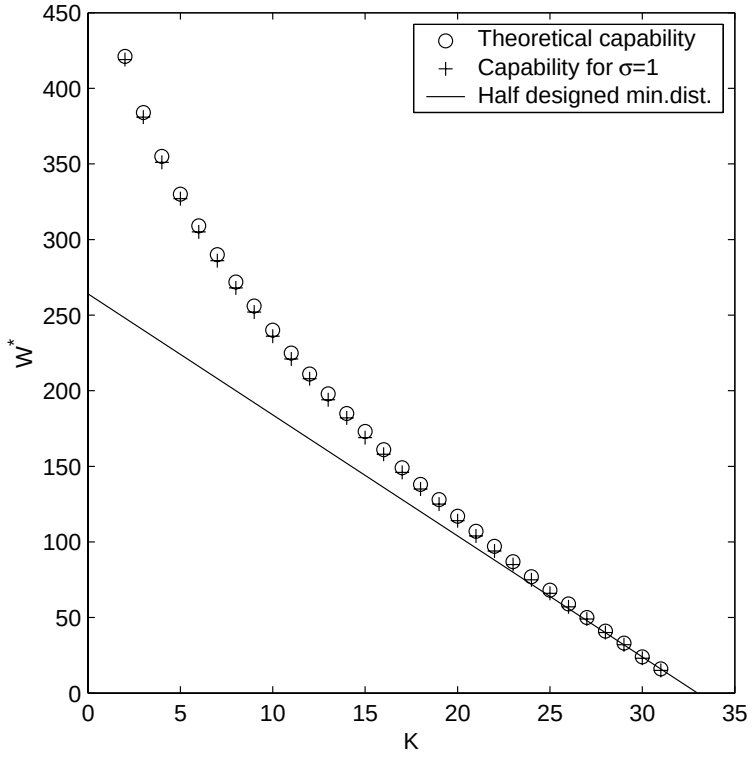

Figure 7.3: *The theoretical (asymptotic) list error-correcting capability of Eqn. 7.18 for concatenated codes where the inner code is a Hadamard code and the outer code is a Reed-Solomon code is compared to the lower bound found by Algorithm 7.26 in the case of Example 7.31. Half the designed minimum distance (which equals the actual minimum distance in this case) is shown for comparison.*

and σ "sufficiently" large if W is at most "about" W^* with

$$W^* := \left\lfloor \frac{q-1}{q} q^{2m} (1 - \sqrt{\delta}) \right\rfloor \tag{7.18}$$

where δ satisfies $dD = (1 - 1/q)(1 - \delta)q^{2m}$.

Consider the case where $q = 2$ and $m = 5$. This gives binary concatenated codes with parameters $(1024, 5K, 528 - 16K)$ for $K \in \{1, \ldots, 31\}$. It turns out that for $\sigma \to \infty$ and $R = 16$ the asymptotic list error-correcting capability calculated by Algorithm 7.26 is at most 1 below Eqn. 7.18 and agrees with Eqn. 7.18 for most values of K. Suppose that we choose $\sigma = 1$. Since the inner decoding radius is quite large the multiplicities are still large, so we would expect the error-correcting capability to approach Eqn. 7.18. That this is indeed the case is seen from Fig. 7.3.

It should be mentioned that in this case we have $C_{\max}(R) = 4352$ for all values of K which with the decoding algorithms available at the moment

of writing represents a very significant cost compared to GMD decoding, however, as seen by the figure, the gain in list error-correcting capability is also very significant in many cases.

In order to reduce the decoding cost one may try to use another code or to reduce the decoding radius of the inner code. Consider the case $K = 24$ *which gives a* $(1024, 80, 272)$ *binary code (removing the 0-positions reduces the length to 992). In that case Eqn. 7.18 gives an "asymptotic" list error-correcting capability of* 161 *errors and Algorithm 7.26 gives an error-correcting capability of* 158.

It turns out that when R *is reduced to 10, Algorithm 7.26 gives that* $C_{\max}(R) = 1760$*, but the list error-correcting capability drops to* 145.

As an alternative we try to use as inner code the binary $(15, 6, 6)$ *code with generator matrix* $[IA]$ *where* I *is the* 6 *by* 6 *unity matrix and*

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

and as outer code a $(64, 16, 49)$ *Reed-Solomon code over* $\mathbb{F}_{64}$*. The concatenated code is then a* $(960, 96, \geq 294)$ *binary code so GMD decoding decodes up to 146 errors. The list error-correcting capability of Algorithm 7.24 as calculated by Algorithm 7.26 is shown in Table 7.7 and the results clearly shows that this code performs much better than the Hadamard/Reed-Solomon code above with respect to list error-correcting capability as well as cost of decoding.* □

7.9 Conclusion

A new method — based on Sudan's list decoding algorithm [13] — for error and erasure decoding of a large class of concatenated codes has been presented and the list decoding capability of the method has been determined. Furthermore, the well-known method of decoding concatenated codes using Forney's generalized minimum distance decoding [10] has been

R	σ	W	$C_{\max}(R)$	$\lambda - 1$
3	1	146	384	6
4	1	159	640	8
5	1	166	960	10
6	1	154	1920	15
7/2	2	167	1792	14
4	2	163	2304	17
9/2	2	174	2880	19
5	2	173	3520	21
11/2	2	177	4224	23
6	2	168	6528	29

Table 7.7: List error-correcting capability, W, of Algorithm 7.24 as calculated by Algorithm 7.26 for the $(960, 96, \geq 294)$ code of Example 7.31 for diffent inner decoding radii, R. The decoding cost is indicated by $C_{\max}(R)$ and the number of codewords in the output is at most $\lambda - 1$.

described in a new setup which characterizes a large class of correctable error patterns with weight beyond half the designed distance.

It is remarkable that the optimal choice of reliabilities involved in GMD decoding is identical (up to a constant factor) to the optimal choice of multiplicities in Sudan's algorithm when decoding concatenated codes up to half the designed distance. An interpretation of this is that Sudan's algorithm when used in the way described in Algorithm 7.15 performs a special generalized minimum distance decoding. However, Example 7.22 shows that the output of the methods are not equivalent.

This paper only treats the situation where the outer code is a Reed-Solomon code, however, generalizing the results to having other Sudan-decodable codes as outer codes seems to be straightforward. A less straightforward generalization would be to consider the case of generalized concatenated codes where inner codes with different minimum distances may be used on different positions of the outer word. An example of this situation is the construction of generalized algebraic geometry codes by Xing *et al* [43]. List decoding of these codes will be the subject of a forthcoming paper.

The basic method of this paper does unique decoding of the inner code and list decoding of the outer code, however, generalizing the method of [14] a method involving list decoding of both the inner and the outer code has been described. In that case the error-correcting capability is more difficult

to estimate since it requires (partial) knowledge of the weight distributions of cosets of the inner code. A method for calculating the error-correcting capability by computer has been described. The bottle-neck of the method is to go through all weight distributions of cosets of the inner code which is possible only for relatively short inner codes.

7.10 Appendix

7.10.1 Proof of Theorem 7.16

In Algorithm 7.9 we have

$$C = \sum_{i \in I_{d/2}} \binom{d-2i+1}{2} W_i \tag{7.19}$$

and

$$\hat{Z} = \sum_{i \in I_{d/2}} (d-2i)\hat{W}_i.$$

Observe that $\hat{W}_i = \hat{B}_i$ for $i \leq d/2$ and recall that $W_i = \hat{B}_i + \hat{B}_{d-i}$ and that $\hat{B} = \sum_i i\hat{B}_i$. Then

$$\hat{Z} = dN - \hat{B} - \sum_{i \in I_{d/2}} iW_i.$$

By Theorem 7.10 an error pattern is corrected if $\hat{Z} > \ell$ (where ℓ is given by Eqn. 7.9) which is implied by $\hat{B} < B$ where

$$B = dN - \frac{(\lambda-1)(K-1)}{2} - \sum_{i \in I_{d/2}} \left(i + \frac{(d-2i)(d-2i+1)}{2\lambda} \right) W_i \tag{7.20}$$

with λ given by Eqn. 7.9.

Notice that for any received word B depends only on the W_i's. This means that given the W_i's (which are known after decoding the inner codewords) the algorithm will return any codeword with a difference to the received word that has badness less than B. By finding the worst possible configuration of the W_i's — that is the one which minimizes B — we can find a lower bound on the error-correcting capability (in the sense of list decoding) of the algorithm. For each valid value of λ a lower bound on the minimal value of B is given by the following lemma:

Lemma 7.32
If $\lambda \leq d$ *then*

$$B \geq \frac{d}{2}\left(N - \frac{\lambda(2d+1-\lambda)}{d(d+1)}(K-1)\right). \tag{7.21}$$

If $\lambda \geq d+1$ *then*

$$B \geq \frac{d}{2}\left(N - \frac{(\lambda-1)(2d+2-\lambda)}{d(d+1)}(K-1)\right). \tag{7.22}$$

□

Proof:
The goal is to minimize B (Eqn. 7.20) as a function of the W_i's for any valid value of λ and with the W_i's satisfying the following constraints:

$$\begin{aligned}
\sum_{i \in I_{d/2}} W_i &= N \\
\sum_{i \in I_{d/2}} \binom{d-2i+1}{2} W_i &\geq \lambda(\lambda-1)(K-1) \\
\sum_{i \in I_{d/2}} \binom{d-2i+1}{2} W_i &\leq \lambda(\lambda+1)(K-1) \\
W_i &\geq 0\ , \quad i \in I_{d/2}.
\end{aligned}$$

This is a linear programming problem (for an introduction, see for example [36], especially Ch. 3, Theorem 1 for an optimality criterion). Notice that the third constraint is a slight relaxation of Eqn. 7.9 and we do not require the variables to take on integer values. Therefore, the minimum that we find may be smaller than the actual minimum.

It can be checked that if $\lambda \leq d$ then the minimum is obtained if

$$W_0 = \frac{\lambda(\lambda+1)(K-1)}{d^2+d} \text{ and } W_{d/2} = N - W_0.$$

This implies $W_i = 0$ for $i \notin \{0, d/2\}$ and B then equals the right side of Eqn. 7.21.

If $\lambda \geq d+1$ then the minimum is obtained when

$$W_0 = \frac{\lambda(\lambda-1)(K-1)}{d^2+d} \text{ and } W_{d/2} = N - W_0.$$

In this case B equals the right side of Eqn. 7.22. ■

After this we will see that B is always at least $dD/2$. First consider the case where $\lambda \leq d$. We must show that the coefficent of $K-1$ in the right side of Eqn. 7.21 is at least -1. That is

$$-\frac{\lambda(2d+1-\lambda)}{d(d+1)} \geq -1 \Leftrightarrow$$
$$\lambda^2 - (2d+1)\lambda + d^2 + d \geq 0.$$

This polynomial has the roots d and $d+1$ and is therefore always non-negative since λ is an integer.

For $\lambda \geq d+1$ it must be shown that the coefficent of $K-1$ in the right side of Eqn. 7.22 is at least -1. We have

$$-\frac{(\lambda-1)(2d+2-\lambda)}{d(d+1)} \geq -1 \Leftrightarrow$$
$$\lambda^2 - (2d+3)\lambda + d^2 + 3d + 2 \geq 0.$$

The polynomial $\lambda^2 - (2d+3)\lambda + d^2 + 3d + 2$ has the roots $d+1$ and $d+2$ so it is non-negative for all integers λ.

In all cases, the minimum of B is at least $dD/2$ and thus the theorem follows.

7.10.2 Proof of Algorithm 7.18

The assumption that there are no erasures implies that the $\hat{B}_i$'s can only be non-zero for $i = 0, 1, \ldots, d$ and the W_i's can only be non-zero for $i = 0, 1, \ldots, t$ where $t := (d-1)/2$.

This gives

$$C = \sum_{i=0}^{t} \binom{\sigma(d-2i)+1}{2} (\hat{B}_i + \hat{B}_{d-i}) \tag{7.23}$$

and

$$\hat{Z} = \sigma \sum_{i=0}^{t} (d-2i)\hat{W}_i = \sigma(dN - \hat{B} - \sum_{i=1}^{t} i(\hat{B}_i + \hat{B}_{d-i})). \tag{7.24}$$

Notice that the last sum satisfies

$$0 \leq \sum_{i=1}^{t} i(\hat{B}_i + \hat{B}_{d-i}) \leq \hat{B}.$$

This implies that for a given value of $\hat{B}$ the value of $\hat{Z}/\sigma$ satisfies

$$dN - 2\hat{B} \leq \hat{Z}/\sigma \leq dN - \hat{B}.$$

Suppose that we choose some badness, B, and let the integer Z_σ vary through all the possible values of $\hat{Z}/\sigma$. If we have some way of determining the maximal possible value of C for each value of Z_σ we can compare the corresponding value of ℓ (as given by Eqn. 7.8) — denote this by $\ell(B, Z_\sigma)$ — with σZ_σ. If $\sigma Z_\sigma > \ell(B, Z_\sigma)$ in all cases then any error pattern of badness B is corrected.

Determining the maximal possible value of C for a given badness, B, and a given value of Z_σ can be formulated as the following linear programming problem:

Find

$$C(B, Z_\sigma) := \max \sum_{i=0}^{t} \binom{\sigma(d-2i)+1}{2} (B_i^* + B_{d-i}^*)$$

where the variables $B_0^*, \ldots, B_d^*$ satisfy the constraints

$$\begin{aligned} \sum_{i=0}^{d} B_i^* &= N \\ \sum_{i=0}^{d} i B_i^* &= B \\ \sum_{i=0}^{t} (d-2i) B_i^* &= Z_\sigma \\ B_i^* &\geq 0 \quad \text{for } i = 0, \ldots, d. \end{aligned}$$

When allowing the B_i^*'s to be rational numbers (instead of requiring integer values), it can be checked that an optimal solution is

$$\begin{aligned} B_t^* &= \frac{dN - B - Z_\sigma}{t} \\ B_d^* &= \frac{2B + Z_\sigma - dN}{d} \\ B_0^* &= N - B_t^* - B_d^* \end{aligned} \tag{7.25}$$

This gives

$$\begin{aligned} C(B, Z_\sigma) &= \frac{\sigma d(\sigma d+1)}{2}(N - B_t^*) + \frac{\sigma(\sigma+1)}{2}B_t^* \\ &= \sigma(\sigma(d+1)+1)(B + Z_\sigma) - \frac{\sigma d(\sigma(d+2)+1)}{2}N \end{aligned} \tag{7.26}$$

and $\ell(B, Z_\sigma)$ denotes the corresponding value of ℓ given by Eqn. 7.8 with $C = C(B, Z_\sigma)$.

So if a badness, B, is given and it is found by the above method that $\sigma Z_\sigma > \ell(B, Z_\sigma)$ for all valid Z_σ (that is for $Z_\sigma = dN - 2B, \ldots, dN - B$) then all error patterns of badness equal to B are corrected.

In order to show that all error patterns of smaller badness are corrected as well, assume that $\sigma Z_\sigma > \ell(B, Z_\sigma)$ for $Z_\sigma \in \{dN - 2B, \ldots, dN - B\}$. We must show that $\sigma Z_\sigma > \ell(B-1, Z_\sigma)$ for $Z_\sigma \in \{dN - 2(B-1), \ldots, dN - (B-1)\}$ which implies that all error patterns of badness equal to $B-1$ are corrected.

Let $Z_\sigma \in \{dN - 2(B-1), \ldots, dN - (B-1)\}$ be given. Then $Z_\sigma - 1 \in \{dN - 2B, \ldots, dN - B\}$ and

$$\sigma Z_\sigma > \sigma(Z_\sigma - 1) > \ell(B, Z_\sigma - 1) = \ell(B-1, Z_\sigma).$$

The second ">" follows from the assumption and the "=" follows from the fact that $C(B, Z_\sigma - 1) = C(B-1, Z_\sigma)$ as it is seen in Eqn. 7.26. So all error patterns of badness equal to $B-1$ are corrected. That all error patterns of smaller badness are corrected follows by induction.

7.10.3 Proof of Theorem 7.20

Consider using Algorithm 7.10.2 on a sequence of concatenated codes. Suppose that the badness, B, given as input to the algorithm satisfies $B/N_a \to \beta$ for $a \to \infty$. Furthermore, let B_t^* be given by Eqn. 7.25 and define β_t by $B_t^*/N_a \to \beta_t$ for $a \to \infty$.

Then Eqn. 7.26 gives

$$\frac{C(B, Z_\sigma)}{N\sigma^2} \to \frac{d^2}{2}(1 - \beta_t) + \frac{1}{2}\beta_t = \frac{d^2 - \beta_t(d^2-1)}{2} \text{ for } a \to \infty$$

and Lemma 7.11 gives that

$$\frac{\ell_\sigma(B, Z_\sigma)}{N_a} \to \sqrt{\kappa(d^2 - \beta_t(d^2-1))}.$$

Furthermore, Eqn. 7.24 implies

$$Z_\sigma / N_a = d - t\beta_t - \beta$$

Therefore, $\sigma Z_\sigma > \ell(B, Z_\sigma)$ if

$$d - t\beta_t - \beta > \sqrt{\kappa(d^2 - \beta_t(d^2-1))} \Leftrightarrow \beta < \beta^*$$

where

$$\beta^* := d - t\beta_t - \sqrt{\kappa(d^2 - \beta_t(d^2-1))}. \tag{7.27}$$

In order to find the minimum of β^* as a function of β_t notice that

$$\frac{\partial \beta^*}{\partial \omega_t} = \frac{d-1}{2}\left((d+1)\sqrt{\frac{\kappa}{(d^2 - \beta_t(d^2-1))}} - 1\right)$$

and

$$\frac{\partial \beta^*}{\partial \beta_t} \leq 0 \Leftrightarrow \beta_t \leq \frac{d^2}{d^2-1} - \kappa\frac{d+1}{d-1}$$

Since $0 \leq \beta_t \leq 1$ the value of β^* is minimized for

$$\beta_t := \max\left\{0, \min\left\{1, \frac{d^2}{d^2-1} - \kappa\frac{d+1}{d-1}\right\}\right\}.$$

Inserting in Eqn. 7.27 gives the theorem.

7.10.4 Proof of Algorithm 7.26

For given W and R it follows by definition that $C(W, Z, R)$ is an upper bound on the maximal value of C in the outer decoding of Algorithm 7.24 for $\hat{Z} = Z$ and an error pattern of weight W.

What remains to be shown is that

1. Z goes through all possible values of $\hat{Z}$.
2. When it has been shown that all error patterns of weight equal to W are corrected then it can be concluded that all error patterns of smaller weight are corrected as well.

These are shown below:

1. Besides from showing that Z goes through all possible values of $\hat{Z}$, the following lemma also shows that it suffices to run the algorithm for those values of Z which are divisible by σ divided by the denominator of R.

   Lemma 7.33
   The value of $\hat{Z}$ is limited by the following inequalities:

   $$RN - \hat{W} \leq \hat{Z}/\sigma \leq RN - \hat{W}R/n.$$

   Furthermore, if $R = a/b$ where $\gcd(a, b) = 1$ then σ/b divides $\hat{Z}$. □

   Proof:
   We have

   $$\hat{Z} = \sigma \sum_{i=0}^{\lfloor R \rfloor} (R - i)\hat{W}_i = \sigma(RN - \hat{W} + \sum_{i=\lfloor R \rfloor + 1}^{n} (i - R)\hat{W}_i)$$

   Notice that the last sum in the right-most expression above is never negative. This gives the left inequality of the lemma.

   On the other hand, the same sum is a linear combination with positive coefficients of the $\hat{W}_i$'s so it obtain its maximal value for all $\hat{W}_i$'s equal to zero except for one, $\hat{W}_{\bar{\imath}}$, which equals its maximal value. Each $\hat{W}_i$ satisfies $\hat{W}_i \leq \hat{W}/i$. So the maximal value of the sum is $(1 - R/\bar{\imath})\hat{W}$. This is maximized by choosing $\bar{\imath} = n$ (the maximal valid value of $\bar{\imath}$). Therefore, the maximal value of the sum is $(1 - R/n)\hat{W}$. This gives the right inequality of the lemma.

   The value of $\hat{Z}$ is a sum of integers in the form

   $$\sigma \left(\frac{a}{b} - m \right)$$

   where $m \in \mathbb{N}$. Since each term is divisible by σ/b, so is $\hat{Z}$. ■

2. First, the following lemma is needed:

   Lemma 7.34
   Let $\mathcal{C}_1$ be a linear (n, k) code over $\mathbb{F}_q$ and let $r_j \in \mathbb{F}_q^n$ be an arbitrary word. Then any coset, $r_j + \mathcal{C}_1$ has a word of weight at least $n(q-1)/q$. □

Proof:
Notice that (recall the Plotkin bound):

$$\sum_{c \in \mathcal{C}_1} \mathrm{d}(-r_j, c) = nq^{k-1}(q-1).$$

Let $\bar{c} \in \mathcal{C}_1$ be chosen such that the distance to $-r_j$ is maximal. The maximal distance must be greater than the average distance, so

$$\mathrm{d}(0, r_j + \bar{c}) = \mathrm{d}(-r_j, \bar{c}) \geq n(q-1)/q.$$

■

We then have the following lemma saying that if it is found in Algorithm 7.26 that all error patterns of weight W are corrected then all error patterns of smaller weight are corrected as well as long as $W \leq nN(q-1)/q$, however, this is no limitation since a random word in average has distance $nN(q-1)/q$ to a given codeword so it is not desirable to attempt to correct that many errors.

Lemma 7.35
Let W be given with $W \leq nN(q-1)/q$. Assume that $Z > \ell(W, Z, R)$ for $Z = \sigma(RN - W), \ldots, \sigma(RN - WR/n)$.

Then $Z > \ell(W^*, Z, R)$ for $W^* < W$ and $Z = \sigma(RN - W^*), \ldots, \sigma(RN - W^*R/n)$. □

Proof:
Let $Z \in \{\sigma(RN - (W-1)), \ldots, \sigma(RN - (W-1)R/n)\}$ and suppose that W_i^* for $i = 0, \ldots, n$ are an optimal solution of the linear programming problem of the algorithm when the error weight is $W - 1$. We want to show that $Z > \ell(W-1, Z, R)$.

Since $W - 1 < Nn(q-1)/q$ we must have $W_{\bar{\imath}}^* > 0$ for some $\bar{\imath} < n(q-1)/q$ and since we are really only interested in optimal solutions for integer W_i^*'s we can assume that $W_{\bar{\imath}}^* \geq 1$. By Lemma 7.34 it is possible to choose j such that $j > \bar{\imath}$ and $C_{\max}(j, R) \geq C_{\max}(\bar{\imath}, R)$ (since the — maybe not unique — coset that maximizes $C_{\max}(i, R)$ must have a word of weight at least $n(q-1)/q$). Now, let $\bar{W}_i^* := W_i^*$ for $i \notin \{\bar{\imath}, j\}$ and

$$\bar{W}_{\bar{\imath}}^* := W_{\bar{\imath}}^* - 1/(j - \bar{\imath})$$
$$\bar{W}_j^* := W_j^* + 1/(j - \bar{\imath})$$

Then the $\bar{W}_i^*$'s form a possible (not necessarily optimal) solution of the linear programming problem above for error weight W and some valid $\bar{Z} \leq Z$. The value of the object function is at least the same as for the W_i^*'s so it can be concluded that $C(W-1, Z, R) \leq C(W, \bar{Z}, R)$. Now we have

$$Z \geq \bar{Z} > \ell(W, \bar{Z}, R) \geq \ell(W^*, Z, R)$$

which is the desired result. ■

Acknowledgements

Some of this work was done while visiting the Laboratory for Computer Science at MIT and Bell Labs Innovations at Lucent Technology. Furthermore, the author would like to thank T. Høholdt, M. Sudan, A. Barg, and two anonymous referees for fruitful discussions and helpful comments.

Chapter 8

Decoding Xing-Ling codes

R. Refslund Nielsen[1]

Abstract

Recently, a new construction of linear codes was proposed with a lower bound on the minimum distance which in many cases is the best known for the given length and dimension. In this correspondence an efficient decoding method for error and erasure decoding is developed which corrects all error patterns of weight up to half the designed minimum distance. A list decoding method is proposed and the error correcting capability as well as the asymptotic behaviour of the codes and the list decoding method is described. Furthermore, an extension of the construction is proposed which results in some linear codes with larger minimum distance than known linear codes with the same alphabet, length, and dimension.

Index terms: Decoding, list decoding, Reed-Solomon codes, subfield subcodes, m-metric.

8.1 Introduction

In [42] Xing and Ling describes a new construction of a class of linear codes. The construction is similar to that of Reed-Solomon codes (when constructed as evaluation codes), however, in the new construction elements from an extension field are evaluated in polynomials which are guaranteed to give values in the base field. This permits longer codes than Reed-Solomon codes over the base field, but since the polynomials must satisfy some restrictions the dimension is lower than a corresponding Reed-Solomon code over the extension field.

[1]Department of Mathematics, Technical University of Denmark, Building 303, DK-2800 Lyngby, Denmark. E-mail: R.R.Nielsen@mat.dtu.dk

Xing and Ling prove a lower bound on the minimum distance of their code construction, however, they do not give a decoding method. Since the code is basically a subfield subcode of a Reed-Solomon code over the extension field it is natural to try to decode a received word using a decoder for the Reed-Solomon code. It turns out that this approach often, but not always, corrects all error patterns up to half the minimum distance bound.

In this correspondence a procedure is developed which always corrects any error pattern of weight up to half the minimum distance bound (and some well-defined heavier error patterns). For even alphabet size, q, the procedure involves two Reed-Solomon decodings over $\mathbb{F}_{q^2}$ and one Reed-Solomon decoding over $\mathbb{F}_q$. For odd q the procedure involves one Reed-Solomon decoding over $\mathbb{F}_{q^2}$, one Reed-Solomon decoding over $\mathbb{F}_q$, and one decoding of a Generalized Reed-Solomon m-code which is defined in this correspondence (Reed-Solomon m-codes was first described by Rosenbloom and Tsfasman [32]).

With list decoding a better error correcting capability is achieved at the expense of possibly getting more than one codeword as output. In this correspondence a list decoding method for Xing-Ling codes is obtained by using Guruswami and Sudan's list decoding method for Reed-Solomon codes (see [13] and Chapter 3) as the underlying decoding method. The asymptotic behaviour of the codes and the list decoding algorithm is discussed. In particular, it is found that asymptotically Xing-Ling codes has a fractional minimum distance of at least $1-\sqrt{\kappa}$ where κ is the information rate.

Furthermore, it is proposed to lengthen Xing-Ling codes by evaluating in the point at infinity on the projective line. This is a well-known method of lengthening Reed-Solomon codes and it turns out that when applied to Xing-Ling codes a number of codes over $\mathbb{F}_7$, $\mathbb{F}_8$, and $\mathbb{F}_9$ are obtained which have larger minimum distance than the best known minimum distance found in Brouwer's table [4] for the given alphabet, length, and dimension.

The correspondence is organized as follows: Section II describes the construction of Xing and Ling, the parameters of the codes are proved, and an extension of the construction is described. In Section III some metrics are defined in order to handle erasures transparently in the decoding methods. Furthermore, it is noted that Xing-Ling codes can be seen as subcodes of one or two supercodes. One of these is defined as a General-

ized Reed-Solomon m-code. Section IV describes the decoding procedure and the error correcting capability is proved. In Section V the asymptotic behaviour of Xing-Ling codes is discussed and a list decoding method is described.

The Appendix describes an error and erasure list decoding method for Generalized Reed-Solomon m-codes. The description includes error and erasure decoding and list decoding of Reed-Solomon codes as a special case.

8.2 Xing-Ling codes

The construction is described by Xing and Ling in [42]. Here a summary is given with slightly different notation. Throughout the correspondence, let $\mathbb{F}_q$ denote a finite field with q elements and if A is a set then let $|A|$ denote the number of elements in A.

Furthermore, if A and B are sets or tuples then we will let

$$(A, B) := (\alpha_1, \ldots, \alpha_{|A|}, \beta_1, \ldots, \beta_{|B|})$$

where $\alpha_1, \ldots, \alpha_{|A|}$ and $\beta_1, \ldots, \beta_{|B|}$ are given fixed enumerations of the elements of A and B respectively. The notation will be extended to multiple sets/tuples/elements by letting $(A_1, \ldots, A_m)$ denote the tuple of elements of $A_1, \ldots, A_m$.

If $f \in \mathbb{F}_q[x]$ is a polynomial and A is a set or tuple of elements of some extension field, $\mathbb{F}_{q^r}$, then we let

$$f(A) := (f(\alpha_1), \ldots, f(\alpha_{|A|})) \tag{8.1}$$

where $\alpha_1, \ldots, \alpha_{|A|}$ is a given fixed enumeration of the elements of A.

Recall the definition of Reed-Solomon codes[2]:

Definition 8.1
Let $P \subseteq \mathbb{F}_q$ with $|P| = N$ and let $K < N$ be a positive integer. Then the set

$$RS(P, K) := \{f(P) \mid f \in \mathbb{F}_q[x] \wedge \deg(f) < K\}$$

with $f(P)$ defined as in Eqn. 8.1 is called a Reed-Solomon code. □

[2]The definition of Reed-Solomon codes used here also covers what is sometimes referred to as extended and/or punctured Reed-Solomon codes.

It is well-known that $RS(P, K)$ is a linear code over $\mathbb{F}_q$ of length N, dimension K, and minimum distance $N - K + 1$. The set P can contain at most all elements of $\mathbb{F}_q$ so the length of a Reed-Solomon code is at most q.

Informally, a Xing-Ling code is a subfield subcode of a Reed-Solomon code over $\mathbb{F}_{q^2}$. Where a Reed-Solomon code over $\mathbb{F}_{q^2}$ is obtained by evaluating elements of $\mathbb{F}_{q^2}$ in all polynomials of degree at most $K-1$, a Xing-Ling code is obtained by evaluating certain elements of $\mathbb{F}_{q^2}$ in certain polynomials of degree at most $K-1$. The elements and polynomials are chosen in such a way that a code over $\mathbb{F}_q$ is obtained.

A basis of the polynomials to be evaluated consists of a subset of all monomials and monic binomials which give values in $\mathbb{F}_q$ when evaluated in elements from $\mathbb{F}_{q^2}$. Let

$$e_{i,j} := \begin{cases} x^{qi+j} + x^{qj+i} & \text{for } i < j \\ \delta x^{qi+i} & \text{for } i = j \end{cases}$$

where

$$\delta := \begin{cases} 2 & \text{for } q \text{ odd} \\ 1 & \text{for } q \text{ even.} \end{cases} \tag{8.2}$$

Let V_K (for any integer K) denote the following vector space over $\mathbb{F}_q$:

$$V_K := \text{span}\{e_{i,j} \mid \deg(e_{i,j}) < K\}. \tag{8.3}$$

We then have the following definition of Xing-Ling codes:

Definition 8.2
Let $A \subseteq \mathbb{F}_q$ and $B \subseteq \mathbb{F}_{q^2} \backslash \mathbb{F}_q$ be given such that $\beta^q \notin B$ for all $\beta \in B$ and let K be given such that $V_K \neq V_{K-1}$. Then the set

$$XL(A, B, K) := \{f(A, B) \mid f \in V_K\} \tag{8.4}$$

is a Xing-Ling code. □

The main parameters of Xing-Ling codes are summarized in the following Theorem:

Theorem 8.3 ([42], Theorem 2.5, 2.6, 2.9, and 2.10)
The code $XL(A, B, K)$ satisfies the following:

1. *The code is a linear code over* $\mathbb{F}_q$.

2. *Let the number of elements of* A *and* B *be denoted by*

$$n_A := |A| \quad \textit{and} \quad n_B := |B|.$$

The length of the code is then $n = n_A + n_B$ *and if* $K - 1 = qr + s$ *where* $0 \leq s \leq q$ *then the dimension is* $k = (r(r+1))/2 + s + 1$.

3. *Let*

$$z := \begin{cases} \max\{2(r-1), r+s\} & \textit{if } q \textit{ is odd} \\ \max\{r-1, s\} & \textit{if } q \textit{ is even.} \end{cases} \tag{8.5}$$

Then the minimum distance, d, *satisfies* $d \geq d^*$ *where*

$$d^* := n - \lfloor (K - 1 + \max\{\min\{z, n_A\}, 2n_A - \delta q\})/2 \rfloor \tag{8.6}$$

with δ *given by Eqn. 8.2. Notice that for* q *odd the first term of the* max*-expression is always largest since* $n_A \leq q$.

□

Proof:

1. A monomial x^a gives an element in $\mathbb{F}_q$ when evaluated in any element $\beta \in \mathbb{F}_{q^2}$ if $\beta^a = \beta^{aq}$. Since $\beta = \beta^{q^2}$ this means that $a \equiv aq \pmod{q^2-1}$ which is equivalent to

$$a - aq + i(q^2 - 1) = 0 \Leftrightarrow a = i(q+1)$$

for some integer i. This shows that any monomial in the form

$$x^{i(q+1)} \tag{8.7}$$

for $i \in \mathbb{N}$ gives values in $\mathbb{F}_q$ when evaluated in an element of $\mathbb{F}_{q^2}$.

A binomial $x^a + x^b$ gives an element in $\mathbb{F}_q$ when evaluated in any element $\beta \in \mathbb{F}_{q^2}$ if $\beta^a + \beta^b = (\beta^a + \beta^b)^q$. Since $(\beta^a + \beta^b)^q = \beta^{aq} + \beta^{bq}$ and $\beta = \beta^{q^2}$ there are 2 possibilities for this to be satisfied:

(a)

$$\begin{aligned} a &\equiv aq \pmod{q^2 - 1} \\ b &\equiv bq \pmod{q^2 - 1} \end{aligned}$$

(b)

$$\begin{aligned} a &\equiv bq \pmod{q^2-1} \\ b &\equiv aq \pmod{q^2-1} \end{aligned}$$

The first possibility corresponds to a sum of two monomials as found in Eqn. 8.7. The second possibility has the solution $a = qi + j$ and $b = qj + i$ for integers i and j.

Therefore, all polynomials in V_K (Eqn. 8.3) gives a value in $\mathbb{F}_q$ when evaluated in an element of $\mathbb{F}_{q^2}$ and thus the codewords of $XL(A, B, K)$ have elements in $\mathbb{F}_q$. Linearity follows from the facts that V_K is a vector space over $\mathbb{F}_q$ and the evaluation map is linear.

2. The length follows by the definition of the code (Eqn. 8.4).

   The dimension is found by counting the number of elements in the basis of V_K in Eqn. 8.3. Notice that these elements are, indeed, a basis since the monomials and binomials of the set

   $$\{e_{i,j} \mid 0 \leq i \leq j < q\}$$

   all have different degree and, therefore, are linearly independent.

   Let r be the largest integer such that $\deg(e_{0,r}) \leq K - 1$ and write $K - 1$ as $K - 1 = qr + s$ with $0 \leq s < q$. Then

   $$V_K = \text{span}\{e_{i,j} \mid 0 \leq i \leq j \leq r - 1 \vee (j = r \wedge i \leq \min\{s, r\})\}.$$

   The assumption that $V_K \neq V_{K-1}$ implies that $s \leq r$ and $\deg(e_{s,r}) = K - 1$. So the dimension of $XL(A, B, K)$ is

   $$\dim(V_K) = \frac{r(r+1)}{2} + s + 1.$$

3. Let $f \in V_K \backslash \{0\}$. For any $\beta \in \mathbb{F}_{q^2}$ and $i \leq j$ we have $e_{i,j}(\beta) = e_{i,j}(\beta^q)$ so if f has a zero, $\beta \in B$, then $\beta^q \notin B$ must also be a zero of f. Therefore, if z_f denotes the number of zeroes f has in $\mathbb{F}_q$ then f can have at most $\lfloor(\deg(f) - z_f)/2\rfloor$ zeroes in B.

   In order to calculate how many zeroes f may have in $\mathbb{F}_q$ notice that if

   $$f = \sum_{0 \leq i \leq j < q} f_{i,j} e_{i,j} \ , \quad f_{i,j} \in \mathbb{F}_q$$

and

$$g = 2 \sum_{0 \le i < j < q} f_{i,j} x^{i+j} + \delta \sum_{0 \le i < q} f_{i,i} x^{2i}. \tag{8.8}$$

then $f \equiv g \pmod{x^q - x}$ so $g(\alpha) = f(\alpha)$ for all $\alpha \in \mathbb{F}_q$. So if $g = 0$ then f has all of $\mathbb{F}_q$ as zeroes.

If q is odd this can be improved since in that case

$$f - (g + g^q)/2 = \sum_{0 \le i < j < q} f_{i,j}(x^{qi+j} + x^{qj+i} - x^{i+j} - x^{q(i+j)}) + \sum_{0 \le i < q} f_{i,i}(2x^{qi+i} - x^{2i} - x^{2qi}) = \sum_{0 \le i < j < q} f_{i,j}(x^{qi} - x^i)(x^j - x^{qj})$$

The last term is divisible by $(x^q - x)^2$ so $f \equiv (g + g^q)/2 \pmod{(x^q - x)^2}$. This means that if $g = 0$ and q is odd then f has all of $\mathbb{F}_q$ as zeroes of multiplicity 2.

In any case, if $g = 0$ then any $\alpha \in \mathbb{F}_q$ is a zero of multiplicity δ (defined in Eqn. 8.2) of f. Therefore, f can have at most $\max\{0, \lfloor(\deg(f) - \delta q)/2\rfloor\} \le \max\{0, \lfloor(K - 1 - \delta q)/2\rfloor\}$ zeroes in B. In order to avoid cumbersome notation, assume that $K - 1 \ge \delta q$. So if $g = 0$ the weight of $f(A, B)$ satisfies

$$\begin{aligned} \mathbf{w}(f(A, B)) &\ge n - n_A - \lfloor(K - 1 - \delta q)/2\rfloor \\ &= n - \lfloor(K - 1 + 2n_A - \delta q)/2\rfloor. \end{aligned}$$

Now assume that $g \neq 0$. Since $\deg(g) \le \max\{2(r-1), r+s\}$ we have that f has at most $\max\{2(r-1), r+s\}$ zeroes in $\mathbb{F}_q$.

If q is even then this can be improved. For any element $w \in \mathbb{F}_q$ there exists a unique element $\sqrt{w} \in \mathbb{F}_q$ such that $\sqrt{w}^2 = w$ (actually, $\sqrt{w} = w^{q/2}$). Furthermore, Eqn. 8.8 reduces to

$$g = \sum_{0 \le i < q} f_{i,i} x^{2i} = h^2$$

where $h := \sum_{0 \le i < q} \sqrt{f_{i,i}} x^i$. We have that $\deg(h) \le \max\{r - 1, s\}$ and if $\alpha \in \mathbb{F}_q$ then $f(\alpha) = g(\alpha) = h(\alpha)^2$ so f can have at most $\max\{r - 1, s\}$ zeroes among the elements of $\mathbb{F}_q$ if $g \neq 0$.

In all cases, if $g \neq 0$ then f has at most z zeroes in $\mathbb{F}_q$ where z is given by Eqn. 8.5. Therefore, if $g \neq 0$ and z_f denotes the number of zeroes f has in A then $z_f \leq \min\{z, n_A\}$ and

$$\begin{aligned} \mathbf{w}(f(A,B)) &\geq n - z_f - \lfloor (K-1-z_f)/2 \rfloor \\ &= n - \lfloor (K-1+z_f)/2 \rfloor \\ &\geq n - \lfloor (K-1+\min\{z, n_A\})/2 \rfloor. \end{aligned}$$

Combining the lower bounds above on the weight of $f(A,B)$ when $g = 0$ and when $g \neq 0$ gives the lower bound on the minimum distance stated in the theorem.

■

Notice that in the definition of Xing-Ling codes we have the constraints $n_A \leq q$ and $n_B \leq (q^2 - q)/2$ so the length of a Xing-Ling code is at most $q(q+1)/2$. However, it is possible to define an extended Xing-Ling code where the length may be one larger. This is done below.

Suppose that $f \in V_K$ for some K where $V_K \neq V_{K-1}$ and $K - 1 = rq + s$ with $r, s \in \mathbb{N}$ and $s < q$ so $\deg(e_{s,r}) = K - 1$. Then define $f(\infty) \in \mathbb{F}_q$ as the coefficient of $e_{s,r}$ in f. That is

$$f = f(\infty)e_{s,r} + f', \quad f' \in V_{K-1}.$$

We will see $f(\infty)$ as the value obtained by evaluating f in a special element, ∞. This actually corresponds to evaluating the homogenization of f in the point at infinity on the projective line. We may now define Extended Xing-Ling codes as usual Xing-Ling codes with $f(\infty)$ added as an extra position:

Definition 8.4
Let $A \subseteq \mathbb{F}_q$ and $B \subseteq \mathbb{F}_{q^2} \backslash \mathbb{F}_q$ be given such that $\beta^q \notin B$ for all $\beta \in B$ and let K be given such that $V_K \neq V_{K-1}$. Then the set

$$EXL(A, B, K) := \{f(\infty, A, B) \mid f \in V_K\} \tag{8.9}$$

is an Extended Xing-Ling code. □

The main parameters of Extended Xing-Ling codes are given in the following theorem:

Theorem 8.5
The code $EXL(A, B, K)$ satisfies the following:

1. *The code is a linear code over $\mathbb{F}_q$.*
2. *The length of the code is $\bar{n} = n_A + n_B + 1$ and if $K - 1 = qr + s$ where $0 \leq s \leq q$ then the dimension is $k = (r(r+1))/2 + s + 1$.*
3. *Let $\bar{d}$ denote the minimum distance of $EXL(A, B, K)$. If $V_{K-2} = V_{K-1}$ then $\bar{d} \geq d^* + 1$ where d^* is the minimum distance bound (Eqn. 8.6) of $XL(A, B, K-1)$.*

   If $V_{K-2} \neq V_{K-1}$ then $\bar{d} \geq d'$ where d' is the minimum distance bound (Eqn. 8.6) of $XL(A, B, K-2)$.

□

Proof:
1. and 2. are seen as for Xing-Ling codes.
To see 3., let $f \in V_K \backslash \{0\}$.
If $f(\infty) \neq 0$ then

$$\mathbf{w}(f(\infty, A, B)) = 1 + \mathbf{w}(f(A, B)) \geq 1 + d^*$$

since $f(A, B) \in XL(A, B, K)$.

If $f(\infty) = 0$ and $V_{K-2} \neq V_{K-1}$ then $f(A, B) \in XL(A, B, K-1)$ so $\mathbf{w}(f(\infty, A, B)) = \mathbf{w}(f(A, B)) \geq d'$.

If $f(\infty) = 0$ and $V_{K-2} = V_{K-1}$ then $f(A, B) \in XL(A, B, K')$ for some $K' \leq K - 2$ and the minimum distance of this code is clearly at least $d^* + 1$ by Eqn. 8.6. ■

The Extended Xing-Ling code construction results in several improvements to Brouwer's table [4] of best known linear codes (where Xing-Ling codes has already been included). These are listed in Table 8.1.

8.3 Supercodes and extended metrics

The main result of this correspondence is an algorithm for error and erasure decoding of Xing-Ling codes which given a received word finds the sent codeword provided that the weight of the error pattern is less than half the designed minimum distance (the weight of an error pattern containing

q	n_A	n_B	K	n	k	d^*	d_B
7	7	21	24	29	9	15	14
7	7	21	25	29	10	14	13
8	1	32	27	34	9	20	19
8	2	32	37	35	15	15	14
8	2	32	43	35	18	12	11
8	2	32	45	35	20	11	10
8	3	32	37	36	15	15	14
8	3	32	61	36	26	7	6
9	9	36	21	46	6	34	33
9	9	36	30	46	9	29	28
9	9	36	31	46	10	28	27

Table 8.1: Parameters of selected Extended Xing-Ling codes. The minimum distance of each code is lower bounded by d^* while d_B is the best known minimum distance of Brouwer's table for linear codes with the given alphabet size (q), length (n), and dimension (k).

erasures will be defined in this section). The idea behind the algorithm is to see a Xing-Ling code as a code obtained by puncturing a subcode of one or two properly chosen codes, referred to as supercodes. Actually, these supercodes occur implicitly in the proof of the lower bound on the minimum distance of Xing-Ling codes.

All in all, 3 supercodes will be needed. Two of these are properly selected Reed-Solomon codes. The third one is a Generalized Reed-Solomon m-code which will be defined in this section. Decoders of these codes will be subroutines of the decoding algorithm. In the following some metrics will be defined which will allow us to specify the requirements of the subroutines.

8.3.1 Extended Hamming distance

In order to be able to describe decoding with erasures in a transparent way, we extend the Hamming distance to an alphabet containing a symbol, ?, for representing erasures.

First we have the following straight-forward result:

Lemma 8.6
Let $S_1, \ldots, S_N$ be N sets — not necessarily distinct — each with a distance, $\mathrm{d}^{(j)} : S_j^2 \to \mathbb{R}$ for $j = 1, \ldots, N$.

Let $S := S_1 \times \cdots \times S_N$ *and define the function* $\mathrm{d} : S^2 \to \mathbb{R}$ *as follows for* $u, v \in S$*:*

$$\mathrm{d}(u, v) := \sum_{j=1}^{N} \mathrm{d}^{(j)}(u_j, v_j).$$

Then f *is a distance on* S. □

Proof:
For any $u, v, w \in S$ we have

$$\mathrm{d}(u, v) = 0 \Leftrightarrow \sum_{j=1}^{N} \mathrm{d}^{(j)}(u_j, v_j) = 0 \Leftrightarrow \forall j, \mathrm{d}^{(j)}(u_j, v_j) = 0 \Leftrightarrow u = v$$

and

$$\mathrm{d}(u, v) = \sum_{j=1}^{N} \mathrm{d}^{(j)}(u_j, v_j) = \sum_{j=1}^{N} \mathrm{d}^{(j)}(v_j, u_j) = \mathrm{d}(v, u).$$

Finally,

$$\begin{aligned} \mathrm{d}(u, w) + \mathrm{d}(w, v) &= \sum_{j=1}^{N} \mathrm{d}^{(j)}(u_j, w_j) + \mathrm{d}^{(j)}(w_j, v_j) \\ &\geq \sum_{j=1}^{N} \mathrm{d}^{(j)}(u_j, v_j) = \mathrm{d}(u, v). \end{aligned}$$

■

In Section 7.2 the following extension of the Hamming distance is defined:

Let $\mathcal{A}_q := \mathbb{F}_q \cup \{?\}$ and define the function $\mathrm{d}_{\mathcal{A}}^{(1)} : \mathcal{A}_q^2 \to \{0, 1/2, 1\}$ by

$$\mathrm{d}_{\mathcal{A}}^{(1)}(a, b) := \begin{cases} 0 & \text{if } a = b \\ 1/2 & \text{if } a \neq b \text{ and } ? \in \{a, b\} \\ 1 & \text{otherwise.} \end{cases} \tag{8.10}$$

Maybe it seems wrong that $\mathrm{d}_{\mathcal{A}}^{(1)}(?, ?)$ equals 0 instead of $1/2$, however, this is necessary in order to get a distance function. In any case, this is not important to this correspondence since at least one of the arguments of $\mathrm{d}_{\mathcal{A}}^{(1)}$ will always be an element of $\mathbb{F}_q$.

Definition 8.7
For any integer, $n \in \mathbb{N}$, define the function $\mathrm{d}_{\mathcal{A}} : (\mathcal{A}_q^n)^2 \to \mathbb{N}/2$ (where $\mathbb{N}/2 = \{0, 1/2, 1, 3/2, \dots\}$) as follows for $u, v \in \mathcal{A}_q^n$:

$$\mathrm{d}_{\mathcal{A}}(u, v) = \sum_{i=1}^{n} \mathrm{d}_{\mathcal{A}}^{(1)}(u_i, v_i). \tag{8.11}$$

This will be referred to as the extended Hamming distance. We also define the extended Hamming weight of $u \in \mathcal{A}_q^n$ as $\mathbf{w}_{\mathcal{A}}(u) := \mathrm{d}_{\mathcal{A}}(u, 0)$. □

Notice that by Lemma 8.6 the extended Hamming distance is, indeed, a distance on $\mathcal{A}_q^n$. Furthermore, the extended Hamming distance is an extension of the Hamming distance on $\mathbb{F}_q^n$. This means that for any code, $C \subseteq \mathbb{F}_q^n$, the minimum extended Hamming distance equals the minimum Hamming distance. In particular, it is possible to do unique decoding up to half the minimum extended Hamming distance.

8.3.2 Generalized Reed-Solomon m-codes

The construction of Reed-Solomon m-codes is based on a generalized evaluation of polynomials. Usual polynomial evaluation can be described as follows where $\bar{x} \in \mathbb{F}_q$ and $f \in \mathbb{F}_q[x]$ with $\deg(f) < K$. Let f be written as

$$f = \sum_{j=0}^{K-1} f_{\bar{x},j}(x - \bar{x})^j, \quad f_{\bar{x},j} \in \mathbb{F}_q. \tag{8.12}$$

Then $f(\bar{x})$ is exactly the first coefficient, $f_{\bar{x},0}$.

For a given positive integer, m, a generalized evaluation is to take the first m coefficients in Eqn. 8.12, that is the vector

$$f(\bar{x}, m) := (f_{\bar{x},0}, \dots, f_{\bar{x},m-1}).$$

The following short notation will be used for a given set $P = \{P_1, \dots, P_N\} \subseteq \mathbb{F}_q$ with $|P| = N$ and a given tuple $M \in \mathbb{N}^N$:

$$f(P, M) := (f(P_1, M_1), \dots, f(P_N, M_N)). \tag{8.13}$$

Using this notation we can define the following generalization of Reed-Solomon m-codes:

Definition 8.8
Let $P = \{P_1, \ldots, P_N\} \subseteq \mathbb{F}_q$ with $|P| = N$ and let $M = (M_1, \ldots, M_N) \in \mathbb{N}^N$ be given as well as a positive integer, $K < \sum_{j=1}^{N} M_j$. Then the set

$$RS(P, M, K) := \{f(P, M) \mid f \in \mathbb{F}_q[x] \wedge \deg(f) < K\}$$

where $f(P, M)$ is given by Eqn. 8.13 is called a Generalized Reed-Solomon m-code. □

Notice that if $M = (m, \ldots, m)$ for a fixed integer m then a Reed-Solomon m-code is obtained. If $m = 1$ then this is a Reed-Solomon code.

In order to describe the distance properties of Generalized Reed-Solomon m-codes we need the m-distance, [32, Section 1]:

Definition 8.9
Let $m \in \mathbb{N}$ be given and define the m-distance, $\mathrm{d}_m : (\mathbb{F}_q^m)^2 \to \{0, 1, \ldots, m\}$, as follows for any $u, v \in \mathbb{F}_q^m$ where $u = (u_1, \ldots, u_m)$ and $v = (v_1, \ldots, v_m)$:

$$\mathrm{d}_m(u, v) = \begin{cases} 0 & \text{if } u = v \\ m + 1 - j & \text{otherwise, where } j = \min\{j \mid u_j \neq v_j\}. \end{cases} \tag{8.14}$$

Let $M \in \mathbb{N}^N$ and let $S_M := \mathbb{F}_q^{M_1} \times \cdots \times \mathbb{F}_q^{M_N}$. Then define the M-distance, $\mathrm{d}_M : S_M^2 \to \mathbb{N}$, as follows for any $U, V \in S_M$ where $U = (U_1, \ldots, U_N)$ and $V = (V_1, \ldots, V_N)$:

$$\mathrm{d}_M(U, V) := \sum_{j=1}^{N} \mathrm{d}_{m_j}(U_j, V_j). \tag{8.15}$$

□

That d_m is a distance on $\mathbb{F}_q^m$ is shown in Theorem 6.5 and thus is follows by Lemma 8.6 that d_M is a distance on S_M.

The following proposition gives the minimum M-distance of Generalized Reed-Solomon m-codes:

Proposition 8.10
Let $RS(P, M, K)$ be a Generalized Reed-Solomon m-code. Then

$$\min\{\mathrm{d}_M(u, v) \mid u, v \in RS(P, M, K) \wedge u \neq v\} = \sum_{j=1}^{N} M_j - K + 1$$

where d_M is given by Eqn. 8.15. □

Proof:
Notice that the length of $RS(P, M, K)$ is $\sum_{j=1}^{N} M_j$. The proof is then a straight-forward generalization of the proof of Theorem 6.6. Alternatively, the proposition follows from the error correcting capability of the decoder in the appendix in the case $\lambda = 2$ and $s = 1$. ■

8.3.3 Extended m-distance

This subsection describes an extension of the m-distance which makes it possible to describe decoding with erasures under the m-metric in a transparent way.

It turns out that in order to get a distance with properties convenient for this correspondence, we need to consider the set

$$\mathcal{S}_{q,m} := \{(a_1, \ldots, a_m) \in \mathcal{A}_q^m \mid a_\alpha =? \Rightarrow a_{\alpha+1} =? \text{ for } \alpha = 1, \ldots, m-1\}. \tag{8.16}$$

Furthermore, let $\mathrm{s} : \mathcal{S}_{q,m} \to \{1, \ldots, m+1\}$ be defined as follows for any $a = (a_1, \ldots, a_m) \in \mathcal{S}_{q,m}$:

$$\mathrm{s}(a) := \begin{cases} m+1 & \text{if } a \in \mathbb{F}_q^m \\ \alpha & \text{if } a \notin \mathbb{F}_q^m \text{ and } \alpha = \min\{\alpha \mid a_\alpha =?\}. \end{cases} \tag{8.17}$$

Definition 8.11
Define the extended m-distance, $\mathrm{d}_{\mathcal{S},m} : (\mathcal{S}_{q,m})^2 \to \{0, 1/2, \ldots, m\}$, as follows for $a, b \in \mathcal{S}_{q,m}$ and $\alpha := \min\{\mathrm{s}(a), \mathrm{s}(b)\}$, $\beta := \max\{\mathrm{s}(a), \mathrm{s}(b)\}$, $j := \min\{j \mid a_j \neq b_j\}$:

$$\mathrm{d}_{\mathcal{S},m}(a, b) = \begin{cases} 0 & \text{if } a = b \\ \alpha - j + \frac{\beta-\alpha}{2} = \frac{\mathrm{S}(a)+\mathrm{S}(b)}{2} - j & \text{if } a \neq b. \end{cases} \tag{8.18}$$

where $\mathrm{s}(\cdot)$ is defined in Eqn. 8.17.

Let $M = (M_1, \ldots, M_N) \in \mathbb{N}^N$ and let $\mathcal{S}_M := \mathcal{S}_{q,M_1} \times \cdots \times \mathcal{S}_{q,M_N}$. Then define the extended M-distance $\mathrm{d}_{\mathcal{S},M} : \mathcal{S}_M^2 \to \mathbb{N}/2$ as follows for $U, V \in \mathcal{S}_M$ where $U = (U_1, \ldots, U_N)$ and $V = (V_1, \ldots, V_N)$:

$$\mathrm{d}_{\mathcal{S},M}(U, V) := \sum_{j=1}^{N} \mathrm{d}_{\mathcal{S},M_j}(U_j, V_j). \tag{8.19}$$

□

The following theorem shows that the extended m-distance is indeed a distance. It then follows by Lemma 8.6 that the extended M-distace is a distance on $\mathcal{S}_M$ and. Furthermore, it is seen to be an extension of d_M, which means that the minimum $\mathrm{d}_{\mathcal{S},M}$-distance of any code equals its minimum d_M-distance.

Theorem 8.12
The map $\mathrm{d}_{\mathcal{S},m}$ given by Eqn. 8.18 is a distance on $\mathcal{S}_{q,m}$. □

Proof:
The proof has three parts. Let $a, b, c \in \mathcal{S}_{q,m}$ be arbitrary. Then it must be shown that

1. $\mathrm{d}_{\mathcal{S},m}(a,b) = 0 \Leftrightarrow a = b$.
   "$\Leftarrow$" follows from the definition. Suppose that $\mathrm{d}_{\mathcal{S},m}(a,b) = 0$ and $a \neq b$. Then $j = (\alpha+\beta)/2$ and since $j \leq \alpha \leq \beta$ this implies that $j = \alpha = \beta$ so $a = b$ which is a contradiction. This gives "$\Rightarrow$".

2. $\mathrm{d}_{\mathcal{S},m}(a,b) = \mathrm{d}_{\mathcal{S},m}(b,a)$.
   This follows since a and b occur symmetrically in the definition.

3. $\mathrm{d}_{\mathcal{S},m}(a,b) \leq \mathrm{d}_{\mathcal{S},m}(a,c) + \mathrm{d}_{\mathcal{S},m}(c,b)$.
   This is clear if a, b, and c are not pairwise distinct. So suppose that a, b, and c are pairwise distinct and let $j(u,v) = \min\{j \mid u_j \neq v_j\}$ for $u, v \in \{a, b, c\}$. Then

   $$\mathrm{d}_{\mathcal{S},m}(a,b) \leq \mathrm{d}_{\mathcal{S},m}(a,c) + \mathrm{d}_{\mathcal{S},m}(c,b) \Leftrightarrow \mathrm{s}(c) \geq j(a,c) + j(c,b) - j(a,b).$$

   Since $a, b \neq c$ we must have $\mathrm{s}(c) \geq \max\{j(a,c), j(c,b)\}$. Furthermore, $\min\{j(a,c), j(c,b)\} \leq j(a,b)$ since $a_{j(a,b)} \neq b_{j(a,b)}$ so either $a_{j(a,b)} \neq c_{j(a,b)}$ or $c_{j(a,b)} \neq b_{j(a,b)}$. Therefore,

   $$\begin{aligned} \mathrm{s}(c) &\geq \max\{j(a,c), j(c,b)\} \\ &\geq \max\{j(a,c), j(c,b)\} + \min\{j(a,c), j(c,b)\} - j(a,b) \\ &= j(a,c) + j(c,b) - j(a,b). \end{aligned}$$

■

8.4 Decoding

In this section it is assumed a Xing-Ling code, $XL(A,B,K)$, is used for communication and that a codeword $\hat{c} = \hat{f}(A,B) \in XL(A,B,K)$ is sent.

The codeword has the structure $\hat{c} = (\hat{c}_A, \hat{c}_B)$ where $\hat{c}_A = \hat{f}(A)$ (these elements will be referred to as the A-positions) and $\hat{c}_B = \hat{f}(B)$ (these are the B-positions).

Furthermore, a word $r \in \mathcal{A}_q^n$ is received. The goal is to reconstruct the polynomial $\hat{f}$ given r. The word r is also decomposed into two blocks, $r = (r_A, r_B)$ where r_A are the received values on the A-positions and r_B are the received values on the B-positions.

The following notation will be used. Let $C \subseteq \mathbb{F}_q^n$ denote an evaluation code (in this context this just means that there is a one-to-one correspondence between codewords and a set of polynomials). Let $r \in \mathcal{A}_q^n$ be a received word. Then $Decode(C, r) \in \mathbb{F}_q[x] \cup \{\text{“failure”}\}$ denotes the result of some decoding method for C. The result is either a polynomial corresponding to a codeword (presumably close to the received word in a suitable metric) or “failure” which will mean that the method could not make any sense of the received word.

Let $dist : \mathcal{A}_q^n \times A_q^n \to \mathbb{R}$ be a distance function and let $t \in \mathbb{R}$ be fixed. If it for any word, $r \in \mathcal{A}_q^n$ is satisfied that

$$(\exists c \in C : dist(r, c) < t) \Rightarrow Decode(C, r) = f \tag{8.20}$$

where f is the polynomial corresponding to the codeword c then we will say that the decoder $Decode(C, \cdot)$ has error correcting capability t (in the $dist$-metric).

Notice that if d is the minimum distance (with respect to some metric) of a code C then $t = d/2$ is the maximal error correcting capability that any decoder of C can have in that metric. Therefore, $d/2$ is often referred to as the error correcting capability of the code C.

We make the following assumption:

Assumption 8.13
Let $XL(A, B, K)$ be a Xing-Ling code and let $B^{(q)} := \{\beta^q \mid \beta \in B\}$. Then assume that the following decoders are available (the error correcting capabilities corresponds to half the minimum distance of each code in the given metric):

1. *A decoder for $RS((A, B, B^{(q)}), K)$ with error correcting capability $(n_A + 2n_B - K + 1)/2$ in the $\mathrm{d}_{\mathcal{A}}$-metric.*

2. *A decoder for* $RS(A, z+1)$ *with error correcting capability* $(n_A - z)/2$ *in the* $\mathrm{d}_{\mathcal{A}}$*-metric.*

3. *If* q *is odd, a decoder for* $RS((\mathbb{F}_q, B, B^{(q)}), M_{211}, K)$ *where* $M_{211} = (2, \ldots, 2, 1, \ldots, 1)$ *is a tuple containing* q *2's and* $2n_B$ *1's. The error correcting capability must be* $(2q+2n_B-K+1)/2$ *in the* $\mathrm{d}_{S,M_{211}}$*-metric.*

   If q *is even, a decoder for* $RS((\mathbb{F}_q, B, B^{(q)}), K)$ *with error correcting capability* $(q + 2n_B - K + 1)/2$ *in the* $\mathrm{d}_{\mathcal{A}}$*-metric.*

□

We then have the following decoding method (when q is even, $\sqrt{r_A}$ is the vector containing the square root of each element of r_A, furthermore, $M_2 := (2, \ldots, 2) \in \mathbb{N}^q$):

Algorithm 8.14
Input: Xing-Ling code in use, $XL(A, B, K)$*. Received word,* $r \in \mathcal{A}_q^n$*.*
Output: If $f \in V_K$ *exists such that* $\mathrm{d}_{\mathcal{A}}(r, f(A, B)) < d^*/2$ *then* f *is returned.*

$f \leftarrow Decode(RS((A, B, B^{(q)}), K), (r_A, r_B, r_B))$
if $(f \notin V_K$ **or** $\mathrm{d}(f(A, B), r) \geq d^*/2)$ **and** $n_A > z$ **then**
  if q *is odd*
    $g \leftarrow Decode(RS(A, z+1), r_A)$
    $h \leftarrow (g + g^q)/2$
    $f \leftarrow Decode(RS((\mathbb{F}_q, B, B^{(q)}), M_{211}, K), (h(\mathbb{F}_q, M_2), r_B, r_B))$
  else
    $g \leftarrow Decode(RS(A, z+1), \sqrt{r_A})$
    $h \leftarrow g^2$
    $f \leftarrow Decode(RS((\mathbb{F}_q, B, B^{(q)}), K), (h(\mathbb{F}_q), r_B, r_B))$
  end
end
return f □

Notice that any set of subdecoders satisfying Assumption 8.13 can be used in the algorithm above. In particular, the decoder described in the appendix can be applied to all of the codes in Assumption 8.13 and the error

correcting capability satisfy the assumption in each case (choose $\lambda = 2$ and $s = 1$).

Notice also that if $r \in \mathbb{F}_q^n$ (that is, if there are no erasures) then the subdecoders need not be able to handle erasures.

The following theorem gives the correctness of Algorithm 8.14 under Assumption 8.13.

Theorem 8.15
Let $XL(A, B, K)$ be a Xing-Ling code used for communication, let $\hat{c}$ be a sent codeword, and let $r \in \mathcal{A}_q^n$ be the received word. If Assumption 8.13 is satisfied and $\mathrm{d}_{\mathcal{A}}(\hat{c}, r) < d^*/2$ then Algorithm 8.14 returns $\hat{f} \in V_K$ such that $\hat{f}(A, B) = \hat{c}$. □

Proof:
For analysis, let $\hat{W}_A := \mathrm{d}_{\mathcal{A}}(r_A, \hat{c}_A)$ and $\hat{W}_B := \mathrm{d}_{\mathcal{A}}(r_B, \hat{c}_B)$ denote the weight of the error pattern in the A-positions and B-positions respectively.

Let $r^* := (r_A, r_B, r_B)$. We have that

$$\hat{c}^* := \hat{f}(A, B, B^{(q)}) = (\hat{c}_A, \hat{c}_B, \hat{c}_B) \in RS((A, B, B^{(q)}), K)$$

and $\mathrm{d}_{\mathcal{A}}(\hat{c}^*, r^*) = \hat{W}_A + 2\hat{W}_B$ so by Assumption 8.13.1,

$$Decode(RS((A, B, B^{(q)}), K), r^*) = \hat{f}$$

if

$$\hat{W}_A + 2\hat{W}_B < (n_A + 2n_B - K + 1)/2 \tag{8.21}$$

Now there are 3 cases:

- Assume that $n_A \leq z$ (with z given as in Eqn. 8.5). Since $n_A \leq q$ we have $n_A \geq 2n_A - q$ and thus

  $$d^* = n - \lfloor (K - 1 + n_A)/2 \rfloor.$$

  On the other hand, $2(\hat{W}_A + \hat{W}_B) = 2\mathrm{d}_{\mathcal{A}}(\hat{c}, r)$ and d^* are integers and $\hat{W}_A \geq 0$ so

  $$\begin{aligned} 2\mathrm{d}_{\mathcal{A}}(\hat{c}, r) < d^* &\Rightarrow \\ 2(\hat{W}_A + \hat{W}_B) < n - (K - 1 + n_A)/2 &\Rightarrow \\ 2(\hat{W}_A + \hat{W}_B) < W_A + n - (K - 1 + n_A)/2 & \end{aligned}$$

which is equivalent to Expression 8.21. Therefore, if $\mathrm{d}(\hat{c}, r) < d^*/2$ then $Decode(RS((A, B, B^{(q)}), K), r^*) = \hat{f}$. Notice that error patterns beyond half the minimum distance bound can be corrected as long as sufficiently many of the errors/erasures are in the A-positions.

- Assume that $n_A > z$ and $\hat{W}_A \geq (n_A - z)/2$. The first assumption gives

$$d^* = n - \lfloor (K - 1 + \max\{z, 2n_A - \delta q\})/2 \rfloor.$$

Furthermore, we have

$$\begin{aligned} 2(\hat{W}_A + \hat{W}_B) < d^* \Rightarrow \\ 2(\hat{W}_A + \hat{W}_B) < n - (K - 1 + \max\{z, 2n_A - \delta q\})/2 \Rightarrow \\ \hat{W}_A + 2\hat{W}_B < n - (K - 1 + z)/2 - (n_A - z)/2 \Leftrightarrow \\ \hat{W}_A + 2\hat{W}_B < n - (K - 1 + n_A)/2 \end{aligned}$$

which is equivalent to Expression 8.21. Therefore, if $\mathrm{d}(\hat{c}, r) < d^*/2$ then $Decode(RS((A, B, B^{(q)}), K), r^*) = \hat{f}$. Heavier error patterns may be corrected if many errors are in the A-positions or if $2n_A - \delta q > z$.

- Assume that $n_A > z$ and $\hat{W}_A < (n_A - z)/2$. By the proof of The. 8.3.3 it is seen that if q is odd then $\hat{c}_A \in RS(A, z + 1)$ and if q is even then $\sqrt{\hat{c}_A} \in RS(A, z + 1)$. So by the assumption on $\hat{W}_A$ it is possible to reconstruct $\hat{c}_A$. We consider the cases q odd and q even separately:

  Suppose that q is odd. Let $g := Decode(RS(A, z + 1), r_A)$ and $h := (g + g^q)/2$. Since $\hat{W}_A < (n_A - z)/2$, Assumption 8.13.2 implies that $\hat{f} \bmod (x^q - x)^2 = h \bmod (x^q - x)^2$. This means that $\hat{f}(\mathbb{F}_q, M_2) = h(\mathbb{F}_q, M_2)$. Now we have

$$\begin{aligned} 2(\hat{W}_A + \hat{W}_B) < d^* \Rightarrow \\ 2(\hat{W}_A + \hat{W}_B) < n - (K - 1 + \max\{z, 2n_A - 2q\})/2 \Rightarrow \\ 2\hat{W}_B < n - (K - 1 + 2n_A - 2q)/2 \Leftrightarrow \\ 2\hat{W}_B < (2q + 2n_B - K + 1)/2. \end{aligned}$$

  So if $2(\hat{W}_A + \hat{W}_B) < d^*$ then

$$\hat{f} = Decode(RS((\mathbb{F}_q, B, B^{(q)}), M_{211}, K), (h(\mathbb{F}_q, M_2), r_B, r_B))$$

by Assumption 8.13.3. Heavier error patterns will often be corrected as well since $z > 2n_A - 2q$.

Suppose that q is even. Let $g := Decode(RS(A, z+1), \sqrt{r_A})$ and $h := g^2$. Assumption 8.13.3 and $\hat{W}_A < (n_A - z)/2$ implies that $\hat{f}(\mathbb{F}_q) = h(\mathbb{F}_q)$. Furthermore,

$$\begin{aligned} 2(\hat{W}_A + \hat{W}_B) < d^* &\Rightarrow \\ 2(\hat{W}_A + \hat{W}_B) < n - (K - 1 + \max\{z, 2n_A - q\})/2 &\Rightarrow \\ 2\hat{W}_B < n - (K - 1 + 2n_A - q)/2 &\Leftrightarrow \\ 2\hat{W}_B < (q + 2n_B - K + 1)/2. & \end{aligned}$$

So if $2(\hat{W}_A + \hat{W}_B) < d^*$ then

$$\hat{f} = Decode(RS((\mathbb{F}_q, B, B^{(q)}), K), (h(\mathbb{F}_q), r_B, r_B))$$

by Assumption 8.13.3.

■

The following gives an example of using Algorithm 8.14.

Example 8.16
Let β denote a primitive element of $\mathbb{F}_{16}$ satisfying $\beta^4 + \beta + 1 = 0$ and let $\alpha = \beta^5$. Then α is a primitive element of $\mathbb{F}_4$ and $\alpha^2 + \alpha + 1 = 0$. Let

$$A := \{0, 1, \alpha\} \quad and \quad B := \{\beta, \beta^2, \beta^3, \beta^6, \beta^7, \beta^{11}\}.$$

Then $XL(A, B, 10)$ is a Xing-Ling code over $\mathbb{F}_4$ of length 9, dimension 5, and minimum distance 4 which is the best possible for a linear code of that alphabet, length, and dimension.

Let

$$\hat{f} := \alpha + x^5 + \alpha^2(x^8 + x^2) + (x^9 + x^6).$$

The corresponding codeword is

$$\hat{c} := \hat{f}(A, B) = (\alpha, \alpha^2, 1, 1, 0, 1, \alpha^2, 1, \alpha).$$

Suppose that $\hat{c}$ is a sent and the following is received:

$$r = (r_A; r_B) := (\alpha, \alpha^2, 1; 1, 1, 1, ?, 1, \alpha).$$

Notice that $\mathrm{d}_{\mathcal{A}}(\hat{c}, r) = 3/2 < 4/2$ *so the error pattern is correctable. We use Algorithm 8.14 as follows where the algorithm of the Appendix is used for the subdecoders.*

First we try to decode the word

$$r^* := (r_A; r_B, r_B) = (\alpha, \alpha^2, 1; 1, 1, 1, ?, 1, \alpha; 1, 1, 1, ?, 1, \alpha)$$

using the code $RS((A, B, B^{(4)}), 10)$ *where* $B^{(4)} := \{\gamma^4 \mid \gamma \in B\}$ *(this is a* $(15, 10, 6)$ *code over* $\mathbb{F}_{16}$*). We let* $\lambda = 2$ *and* $\sigma = 1$ *and get* $T = 3$*. Since* $\mathrm{d}_{\mathcal{A}}(r^*, \hat{f}(A, B, B^{(4)}) = 3 \not< T$ *it cannot be expected to recover* $\hat{f}$ *and indeed a valid polynomial* Q *is*

$$\begin{aligned} Q =& (\beta^4 + \beta^4 x^2 + \beta^4 x^4 + \beta^{14} x^5 + \beta^4 x^6 + \beta^{14} x^7 + \beta^{14} x^9 + \beta^4 x^{10} + \\ & \beta^9 x^{11}) + (\beta^{14} + \beta^{14} x) y \end{aligned}$$

which do not have $y - \hat{f}$ *as a factor.*

Therefore, we proceed and decode $\sqrt{r_A} = (\alpha^2, \alpha, 1)$ *using* $RS(A, 2)$ *which is a* $(3, 2, 2)$ *code over* $\mathbb{F}_4$*. It is seen that if* $g = x + \alpha^2$ *then* $\sqrt{r_A} = g(A)$*. Therefore, let* $h = g^2 = x^2 + \alpha$ *and let*

$$\bar{r} := (h(\mathbb{F}_4); r_B; r_B) = (\alpha, \alpha^2, 1, 0; 1, 1, 1, ?, 1, \alpha; 1, 1, 1, ?, 1, \alpha).$$

This is decoded using the code $RS((\mathbb{F}_4, B, B^{(4)}), 10)$ *which is a* $(16, 10, 7)$ *code over* $\mathbb{F}_{16}$*. We let* $\lambda = 2$ *and* $\sigma = 1$ *and obtain the following valid polynomial* Q*:*

$$\begin{aligned} Q =& (\beta + \beta^{11} x + \beta^{11} x^2 + \beta x^3 + \beta^6 x^4 + \beta^{11} x^5 + \beta x^6 + \beta x^7 + \beta x^8 + \beta^6 x^9 + \\ & \beta^{11} x^{11}) + (\beta^{11} + \beta^6 x + \beta^{11} x^2) y \end{aligned}$$

which has $y - \hat{f}$ *as a factor. Thus we succeed in correcting the given error pattern.* □

The following example illustrates how the correct codeword may be found even though the weight of the error pattern equals half the actual minimum distance of the Xing-Ling code in use.

Example 8.17
Consider a Xing-Ling code $XL(A, B, K)$ *where* $q = 7$*,* $n_A = 7$*,* $n_B = 21$*, and* $K = 17$*. Then* $r = s = 2$ *so* $z = 4$ *(Eqn. 8.5) and the dimension is* $k = 6$*. The lower bound on the minimum distance is* $d^* = 18$*. Given a received*

word, $r \in \mathcal{A}_q^{28}$, Algorithm 8.14 finds the polynomial, $\hat{f}$, corresponding to the sent codeword, $\hat{c}$, where $\hat{c} = \hat{f}(A, B)$, if the weight of the error pattern, $\mathrm{d}_{\mathcal{A}}(\hat{c}, r)$, is less than 9, that is at most $17/2$.

However, suppose that erasures cannot occur and let $\hat{W}_A$ and $\hat{W}_B$ denote the weight of the error pattern in the A and B-positions respectively. Since erasures cannot occur the weight of the error patterns are integers. Then

$$Decode(RS((A, B, B^{(q)}), K), (r_A, r_B, r_B)) = \hat{f}$$

if $\hat{W}_A + 2\hat{W}_B < (n_A + 2n_B - K + 1)/2 = 33/2$. In particular, if $\hat{W}_A \geq (n_A - z)/2 = 3/2$ (implying that $\hat{W}_A \geq 2$) then any error pattern of weight at most $((33 - 1)/2 + 2)/2 = 9$ is corrected.

If $\hat{W}_A < 3/2$ (implying $\hat{W}_A \in \{0, 1\}$) then

$$Decode(RS((\mathbb{F}_q, B, B^{(q)}), M_{211}, K), (h(\mathbb{F}_q, M_2), r_B, r_B)) = \hat{f}$$

if $2\hat{W}_B < (2q + 2n_B - K + 1)/2 = 20$. Any error pattern of weight at most 9 satisfies this.

This means that any error pattern (without erasures) of weight up to 9 can be decoded. A computer search shows that the true minimum distance of this code is in fact 18. This is not inconsistent with the decoding procedure being able to correct all error-patterns of weight up to 9, because each of the 2 decodings may give a codeword at distance 9 from the received word. So in this case the decoding procedure does list decoding with a maximal list size of 2. □

Decoding of Extended Xing-Ling codes can be done in terms of decoders for Xing-Ling codes. We make the following assumption:

Assumption 8.18
Let $EXL(A, B, K)$ be an Extended Xing-Ling code and let $K' < K$ be maximal such that $V_{K'} \neq V_{K'-1}$. Then assume that the following decoders are available:

1. *A decoder for $XL(A, B, K)$ with error correcting capability $d^*/2$ in the $\mathrm{d}_{\mathcal{A}}$-metric where d^* is the designed minimum distance of $XL(A, B, K)$.*

2. *A decoder for $XL(A, B, K')$ with error correcting capability $d'/2$ in the $\mathrm{d}_{\mathcal{A}}$-metric where d' is the designed minimum distance of $XL(A, B, K')$.*

□

Then we have the following algorithm:

Algorithm 8.19
Input: Extended Xing-Ling code in use, $EXL(A, B, K)$. Received word, $r = (r_\infty, r_A, r_B) \in \mathcal{A}_q^n$.
Output: If $f \in V_K$ exists such that $\mathrm{d}_{\mathcal{A}}(r, f(A, B)) < \bar{d}/2$ where $\bar{d}$ is the designed minimum distance of $EXL(A, B, K)$ then f is returned.
Notation: Let $r, s \in \mathbb{N}$ be given such that $K - 1 = qr + s$ with $0 \leq s \leq q$.

$f \leftarrow Decode(XL(A, B, K), (r_A, r_B))$
if $(f \notin V_K$ **or** $\mathrm{d}_{\mathcal{A}}(f(\infty, A, B), r) \geq \bar{d}/2)$ **and** $r_\infty \neq ?$ **then**
$r' \leftarrow (r_A, r_B) - r_\infty \cdot e_{s,r}(A, B)$
$f' \leftarrow Decode(XL(A, B, K'), r')$
$f \leftarrow f' + r_\infty \cdot e_{s,r}$
end
return f □

Proof:
The correctness of the algorithm under Assumption 8.18 is seen by noticing that if $\hat{c}$ is a sent codeword and the weight of the error pattern, $\mathrm{d}_{\mathcal{A}}(\hat{c}, r)$, is less than $\bar{d}/2$ then the first decoding succeeds if r_∞ is erroneous and the second decoding succeeds otherwise. ■

8.5 List decoding

In this section only decoding with errors will be considered since decoding with erasures cannot be described as transparent as in the case of unique decoding.

Let $C \subseteq \mathbb{F}_q^n$ be a code. Given some integer, t, and a received word $r \in \mathbb{F}_q^n$ the list decoding problem is to calculate the set of all codewords in C of distance at most t to r in some suitable metric. If d is the minimum distance of C and $t < d/2$ then at most one codeword is within distance t from r so the list decoding problem reduces to the usual problem of unique decoding.

It is understood in the problem of list decoding that also some upper bound is known on the size of the output set. Furthermore, we will say

that an error pattern can be decoded (or, alternatively, that decoding is successful) if the correct codeword is in the output set — it need not be the only codeword in the set.

The following notation will be used. Let $C \subseteq \mathbb{F}_q^n$ denote an evaluation code and let $r \in \mathbb{F}_q^n$ be a received word. Then $ListDecode(C, r) \subset \mathbb{F}_q[x]$ denotes the result of some list decoding method for C.

Let $dist : \mathbb{F}_q^n \times F_q^n \to \mathbb{R}$ be a distance function and let $t \in \mathbb{R}$ be fixed. If it for any word, $r \in \mathbb{F}_q^n$ is satisfied that

$$ListDecode(C, r) = \{f \in \mathbb{F}_q^n[x] \mid dist(r, c(f)) < t\} \tag{8.22}$$

where $c(f)$ is the codeword corresponding to the polynomial f then we will say that the list decoder $ListDecode(C, \cdot)$ has error correcting capability t (in the $dist$-metric).

8.5.1 Asymptotic results

When finding asymptotic results on a class of codes, we will assume that a sequence of codes is given with parameters (length, dimension, designed minimum distance) $= (n_a, k_a, d_a)$ for $a \in \mathbb{N}$ such that $n_a \to \infty$ for $a \to \infty$. When describing the results we will drop the subscripts, a, and let it be implicit that the results are for $a \to \infty$. It should be noted that there is no requirement that the alphabet of the codes in the sequence is the same. On the contrary, for all cases in this correspondence the alphabet size will tend to infinity with the code length.

Thus, we may say that if a sequence of (N, K, D) Reed-Solomon codes is given such that the rate $K/N \to \kappa$ for some $\kappa \in \mathbb{R}$ with $0 \leq \kappa \leq 1$ then the fractional minimum distance satisfies

$$D/N \to 1 - \kappa.$$

In [13, Theorem 16] Guruswami and Sudan show that the list decoding algorithm which they propose in the same paper solves the list decoding problem for a (N, K, D) Reed-Solomon code if the number of errors, e, and the number of erasures, s, satisfy

$$e + s < N - \sqrt{(N - s)(K - 1)}.$$

Asymptotically this means that if erasures are not allowed then a fractional number of errors τ can be successfully list decoded if

$$\tau < 1 - \sqrt{\kappa}.$$

In the following we will find similar results for Xing-Ling codes.

The asymptotic minimum distance of Xing-Ling codes is given in the theorem below:

Theorem 8.20
Let an infinite sequence of Xing-Ling codes be given such that

- *The length is of the same order of magnitude as $q^2/2$ where q is the alphabet size. That is $2n/q^2 \to 1$.*
- *The rate $k/n \to \kappa$ for some $\kappa \in \mathbb{R}$ with $0 \leq \kappa \leq 1$.*

Then the fractional designed minimum distance satisfies

$$d^*/n \to 1 - \sqrt{\kappa}.$$

□

Proof:
Consider an arbitrary code in the sequence, $XL(A_a, B_a, K_a)$ over $\mathbb{F}_{q_a}$ with length $n_a = n_A + n_B$ where $n_A = |A_a|$ and $n_B = |B_a|$, dimension $k_a = r_a(r_a+1)/2 + s_a + 1$ where $r_a, s_a \in \mathbb{N}$ are defined by $K_a - 1 = r_a q_a + s_a$ with $0 \leq s_a \leq q_a$.

By the definition of Xing-Ling codes we have that $n_A \leq q_a$ so asymptotically, $n_A/n \to 0$ and $n_B/n \to 1$. Furthermore, $2k/r^2 \to 1$ so $K/n \to 2\sqrt{\kappa}$ and $d^*/n \to 1 - 2(\sqrt{\kappa} + 0)/2 = 1 - \sqrt{\kappa}$. ■

Notice that the first assumption on the sequence of Xing-Ling codes in the theorem says that the Xing-Ling codes, $XL(A, B, K)$, should be constructed with $n_B = |B|$ being essentially maximal — we have that $n_B \leq q(q-1)/2$ by the definition of Xing-Ling codes. This means that the fraction of the code length coming from the set A vanishes ($n_A/n \to 0$ since $n_A \leq q$). Choosing $n_A = 0$ (that is $A = \emptyset$) the decoding algorithm of the last section, Algorithm 8.14, reduces to

$f \leftarrow Decode(RS((A, B, B^{(q)}), K), (r_A, r_B, r_B))$
return f

and, asymptotically, a fractional number of errors τ is successfully decoded if

$$\tau < \frac{1 - \sqrt{\kappa}}{2}$$

since the right side is half the asymptotic fractional designed minimum distance.

However, instead of using a unique decoder we may use a list decoder.

We then have the following theorem giving the asymptotic list error correcting capability of Xing-Ling codes. Notice that the list error correcting capability of the Reed-Solomon list decoder required by the theorem equals the list error correcting capability of the list decoder of Guruswami and Sudan [13].

Theorem 8.21
Let an infinite sequence of Xing-Ling codes be given such that

- *The length is of the same order of magnitude as $q^2/2$ where q is the alphabet size. That is $2n/q^2 \to 1$.*
- *The rate $k/n \to \kappa$ for some $\kappa \in \mathbb{R}$ with $0 \leq \kappa \leq 1$.*
- *For each code, $XL(A_a, B_a, K_a)$, with parameters (n_a, k_a, d_a^*) for $a \in \mathbb{N}$ a list decoder $ListDecode(RS((A_a, B_a, B_a^{(q)}), K_a), \cdot)$ is available with error correcting capability $N_a - \sqrt{N_a(K_a - 1)}$ where $N_a := |A_a| + 2|B_a|$.*

Then a fraction of errors, $t/n \to \tau$, can be successfully list decoded whenever

$$\tau < 1 - \sqrt{\sqrt{\kappa}}.$$

□

Proof:
Notice that $2n/N \to 1$ and recall from the proof of Theorem 8.20 that $K/n \to 2\sqrt{\kappa}$. So asymptotically, the Reed-Solomon list decoder corrects a fraction of errors, $1 - \sqrt{(K-1)/N} \to 1 - \sqrt{\sqrt{\kappa}}$.

Let $r = (r_A, r_B) \in \mathbb{F}_q^n$ is the received word and let $r^* := (r_A, r_B, r_B) \in \mathbb{F}_q^N$ be the word given as input to the Reed-Solomon list decoder.

Suppose that $\hat{c} = (\hat{c}_A, \hat{c}_B)$ was sent and let $\hat{W} := \mathrm{d}(\hat{c}, r)$ denote the error weight. Then r has a fraction of $\hat{W}/n$ errors. Let $\hat{W}^* := \mathrm{d}(\hat{c}^*, r^*)$ where $\hat{c}^* := (\hat{c}_A, \hat{c}_B, \hat{c}_B)$. Suppose that $\hat{W}/n \to \omega$ for some constant ω. Then $\hat{W}^*/N \to 2\omega/2 = \omega$. So, asymptotically, the fraction of errors in r is the same as the fraction of errors in r^*. ■

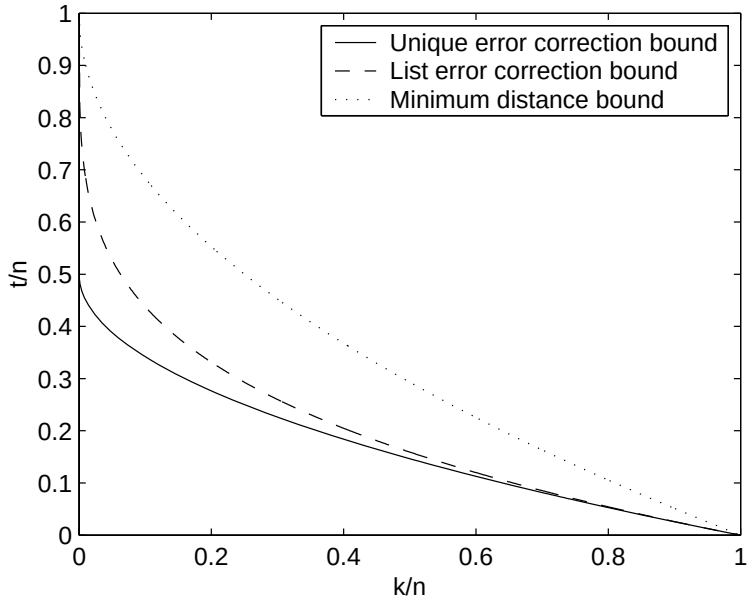

Figure 8.1: *The figure shows the asymptotic fractional designed minimum distance of Xing-Ling codes as a function of the code rate. Furthermore, asymptotic fractional bounds for unique decoding and for list decoding are shown.*

The fractional designed minimum distance of Xing-Ling codes is shown in Figure 8.1 with half the fractional designed minimum distance (which is the fraction of errors that can be corrected by unique decoding) and the list decoding capability obtained in Theorem 8.21.

8.5.2 A list decoding algorithm

For a given code the situation may be different from the asymptotic case since the designed minimum distance may not exactly resemble the asymptotic value. Furthermore, even though the list decoding algorithm of Guruswami and Sudan always can be implemented with an asymptotic running time that is polynomial in the code length, an implementation for a given code often has a computational complexity which is not practical. In that case a faster algorithm can be obtained at the expense of a possibly reduced list error correcting capability.

Therefore, it is desirable to gain as much as possible by carefully choosing how to apply the list decoding algorithm. The natural thing to do is to apply list decoders in Algorithm 8.14.

We make the following assumption:

Assumption 8.22
Let $XL(A, B, K)$ be a Xing-Ling code. Then assume that the following decoders are available for some positive integers λ_1, λ_2, T_1, and T_2 (the size of the output refers to the number of polynomials in the output set):

1. *A list decoder for $RS((A, B, B^{(q)}), K)$ with error correcting capability T_1 in the Hamming metric and output size upper bounded by $\lambda_1 - 1$.*
2. *A decoder for $RS(A, z+1)$ with error correcting capability $(n_A - z)/2$ in the Hamming metric.*
3. *If q is odd, a list decoder for $RS((\mathbb{F}_q, B, B^{(q)}), M_{211}, K)$ where $M_{211} = (2, \ldots, 2, 1, \ldots, 1)$ is a tuple containing q 2's and $2n_B$ 1's. The error correcting capability must be T_2 in the $\mathrm{d}_{\mathcal{S},M_{211}}$-metric and the output size must be upper bounded by $\lambda_2 - 1$..*

   If q is even, a decoder for $RS((\mathbb{F}_q, B, B^{(q)}), K)$ with error correcting capability T_2 in the Hamming metric and output size upper bounded by $\lambda_2 - 1$.

□

We then have the following decoding method (when q is even, $\sqrt{r_A}$ is the vector containing the square root of each element of r_A):

Algorithm 8.23
Input: Xing-Ling code in use, $XL(A, B, K)$. Received word, $r \in \mathbb{F}_q^n$.
Output: Let

$$T := \begin{cases} T_1/2 & \text{if } n_A \leq z \\ \min\{(2T_1 + n_A - z)/4, \max\{T_1/2, T_2/2\}\} & \text{if } n_A > z. \end{cases} \tag{8.23}$$

Then the output is the set

$$\{f \in V_K \mid \mathrm{d}(r, f(A, B)) < T\}.$$

$F \leftarrow ListDecode(RS((A, B, B^{(q)}), K), (r_A, r_B, r_B))$
if $n_A > z$ **then**
  if *q is odd*
    $g \leftarrow Decode(RS(A, z+1), r_A)$
    $h \leftarrow (g + g^q)/2$

$F \leftarrow F \cup ListDecode(RS((\mathbb{F}_q, B, B^{(q)}), M_{211}, K), (h(\mathbb{F}_q, 2), r_B, r_B))$

else

$g \leftarrow Decode(RS(A, z+1), \sqrt{r_A})$

$h \leftarrow g^2$

$F \leftarrow F \cup ListDecode(RS((\mathbb{F}_q, B, B^{(q)}), K), (h(\mathbb{F}_q), r_B, r_B))$

end

end

return F □

The following theorem shows the correctness of the algorithm.

Theorem 8.24
Let $XL(A, B, K)$ be a Xing-Ling code used for communication and suppose that a codeword, $\hat{c} = \hat{f}(A, B)$ is sent, but $r \in \mathbb{F}_q^n$ is received. If Assumption 8.22 is satisfied for some integers λ_1, λ_2, T_1, and T_2 and if $\mathrm{d}(\hat{c}, r) < T$ where T is defined in Eqn. 8.23 then $\hat{f}$ is in the set of polynomials returned by Algorithm 8.23. Furthermore, the number of polynomials in the set is upper bounded by Λ given by

$$\Lambda := \begin{cases} \lambda_1 - 1 & \text{if } n_A \leq z \\ \lambda_1 + \lambda_2 - 2 & \text{if } n_A > z. \end{cases}$$

□

Proof:
For analysis, let $\hat{W}_A := \mathrm{d}_{\mathcal{A}}(r_A, \hat{c}_A)$ and $\hat{W}_B := \mathrm{d}_{\mathcal{A}}(r_B, \hat{c}_B)$ denote the weight of the error pattern in the A-positions and B-positions respectively.

Let $r^* := (r_A, r_B, r_B)$. We have that

$$\hat{c}^* := \hat{f}(A, B, B^{(q)}) = (\hat{c}_A, \hat{c}_B, \hat{c}_B) \in RS((A, B, B^{(q)}), K)$$

and $\mathrm{d}_{\mathcal{A}}(\hat{c}^*, w^*) = \hat{W}_A + 2\hat{W}_B$ so by Assumption 8.22.1,

$$\hat{f} \in ListDecode(RS((A, B, B^{(q)}), K), r^*)$$

if $\hat{W}_A + 2\hat{W}_B < T_1$ which is equivalent to

$$2(\hat{W}_A + \hat{W}_B) < T_1 + \hat{W}_A.$$

Suppose that $2\hat{W}_A < n_A - z$ and $2\hat{W}_B < T_2$. If q is odd then

$$\hat{f} \in ListDecode(RS((\mathbb{F}_q, B, B^{(q)}), M_{211}, K), (h(\mathbb{F}_q, 2), r_B, r_B)).$$

If q is even then

$$\hat{f} \in ListDecode(RS((\mathbb{F}_q, B, B^{(q)}), K), (h(\mathbb{F}_q), r_B, r_B)).$$

In any case, we have that if $2\hat{W}_A < n_A - z$ then $\hat{f}$ is in the output if $2\hat{W}_B < T_2$ or, equivalently,

$$2(\hat{W}_A + \hat{W}_B) < T_2 + 2\hat{W}_A.$$

All in all we have that $\hat{f}$ is in the output if $\mathrm{d}(\hat{c}, r) < T(\hat{W}_A)$ where

$$T(\hat{W}_A) := \begin{cases} \max\{(T_1 + \hat{W}_A)/2, (T_2 + 2\hat{W}_A)/2\} & \text{if } \hat{W}_A < (n_A - z)/2 \\ (T_1 + W_A)/2 & \text{if } \hat{W}_A \geq (n_A - z)/2. \end{cases}$$

So the correctness of the algorithm follows since $T = \min\{T_{\hat{W}_A} \mid \hat{W}_A = 0, \ldots, n_A\}$.

The upper bound on the number of polynomials in the output set is simply the sum of the upper bounds on the number of polynomials in the outputs of the subdecoders. ■

Table 8.2 gives list error-correcting capability of Algorithm 8.23 for selected Xing-Ling codes when the list subdecoders are obtained from the algorithm in the appendix for maximal values of λ and a given value of σ.

8.6 Appendix

In this appendix an algorithm is given for decoding Generalized Reed-Solomon m-codes (Definition 8.8). The algorithm can be efficiently implemented and in the general case it is a list decoding algorithm with error correcting capability strictly greater than half the minimum distance of the code (in the metric associated with the code), however, by choosing the parameters of the algorithm appropriately, an algorithm for unique decoding with error correcting capability equal to half the minimum distance is obtained. The algorithm of this appendix is a modification of Algorithm 6.9 which is a list decoding algorithm for Reed-Solomon m-codes and for errors only.

q	n	k	d^*	σ	$\lfloor\lfloor T_1 \rfloor\rfloor$	$\lfloor\lfloor T_2 \rfloor\rfloor$	$\lfloor\lfloor T \rfloor\rfloor$	Λ
7	22	6	14	2	14	—	7	3
7	24	6	15	2	16	—	8	3
7	28	5	19	2	19	24	10	7
7	28	5	19	3	20	24	11	11
7	28	6	18	1	16	21	9	4
7	28	6	18	2	18	23	10	7
7	28	9	14	2	13	17	7	6
8	28	3	24	1	28	—	14	3
8	34	10	19	3	19	20	10	8
8	34	15	14	3	13	14	7	8
8	34	17	12	7	11	12	6	16
9	45	6	33	3	37	44	20	12
9	45	8	29	2	29	35	16	6
9	45	9	28	2	28	34	15	6
9	45	10	27	3	28	34	15	10
9	45	21	16	5	16	21	8	12

Table 8.2: List error-correcting capability of Algorithm 8.23 for selected Xing-Ling codes constructed with $n_B = q(q-1)/2$ and with minimum distance at least d^*. The notation $\lfloor\lfloor T \rfloor\rfloor$ is used to denote the largest integer which is strictly smaller than T. The number of codewords of distance at most $\lfloor\lfloor T \rfloor\rfloor$ to any word is at most Λ.

Notice that since Reed-Solomon codes are a special case of Generalized Reed-Solomon m-codes, the algorithm can also be used as a (list) decoding algorithm for Reed-Solomon codes. In fact, if the parameters of the algorithm are chosen appropriately, the algorithm reduces to Guruswami and Sudan's list decoding algorithm [13].

8.6.1 Bivariate polynomials

First, some results on bivariate polynomials are needed.

Let $\alpha, \beta \in \mathbb{N}$ be integers. Then the (α, β) weighted degree of some monomial, $x^a y^b \in \mathbb{F}_q[x, y]$ is defined as

$$\deg^{(\alpha,\beta)}(x^a y^b) := \alpha a + \beta b.$$

This is extended to all non-zero bivariate polynomials as follows. Let $Q \in \mathbb{F}_q[x, y]$ with

$$Q = \sum_{a,b \in \mathbb{N}} Q_{a,b} x^a y^b, \quad Q_{a,b} \in \mathbb{F}_q.$$

Then

$$deg^{(\alpha,\beta)}(Q) := \max\{\alpha a + \beta b \mid Q_{a,b} \neq 0\}.$$

We then have the following lemma generalizing Lemma 3.4:

Lemma 8.25
Let $K \geq 2$ and $i \geq K - 1$ be given integers. Furthermore, let λ be given such that $2 \leq \lambda \leq \lambda_{\max}$ where $\lambda_{\max}$ is the integer satisfying the inequality

$$\binom{\lambda_{\max}}{2} \leq \frac{i-1}{K-1} < \binom{\lambda_{\max}+1}{2}. \tag{8.24}$$

Let $\phi_1, \phi_2, \ldots$ be an enumeration of all monomials of $\mathbb{F}_q[x, y]$ where the degree in y is less than λ. So

$$\{\phi_1, \phi_2, \ldots\} = \{x^a y^b \in \mathbb{F}_q[x, y] \mid b < \lambda\}.$$

If the $(1, K-1)$ weighted degree of the enumeration is increasing, that is if

$$\deg^{(1,K-1)}(\phi_1) \leq \deg^{(1,K-1)}(\phi_2) \leq \cdots$$

then

$$\deg^{(1,K-1)}(\phi_i) = \left\lfloor \frac{i-1}{\lambda} + \frac{(\lambda-1)(K-1)}{2} \right\rfloor.$$

□

Proof:
For $j = 0, \ldots, \lambda - 1$ and $\ell \in \mathbb{N}$ define

$$M_j(\ell) := \{x^a y^j \mid \deg^{(1,K-1)}(x^a y^j) < \ell\}.$$

Then we have

$$\begin{aligned}
&\sum_{j=0}^{\lambda-1} |M_j(\ell)| < i \Leftrightarrow \\
&\lambda\ell - \lambda(\lambda-1)(K-1)/2 < i \Leftrightarrow \\
&\ell \leq (i-1)/\lambda + (\lambda-1)(K-1)/2.
\end{aligned}$$

The lemma follows since $\deg^{(1,K-1)}(\phi_i)$ must equal the maximal ℓ satisfying the inequality above. ■

Furthermore, we have the following definition:

Definition 8.26
Let $(\bar{x}, \bar{y}) \in \mathbb{F}_q \times \mathbb{F}_q[x]$ and let $Q \in \mathbb{F}_q[x, y] \backslash \{0\}$. Then let Q be written as

$$Q = \sum_{a,b} Q_{a,b}^{(\bar{x},\bar{y})} (x - \bar{x})^a (y - \bar{y})^b, \quad Q_{a,b}^{(\bar{x},\bar{y})} \in \mathbb{F}_q.$$

Let $\sigma, u \in \mathbb{N}$ be given integers. If

$$Q_{a,b}^{(\bar{x},\bar{y})} = 0 \text{ for all } a, b \in \mathbb{N} \text{ with } a/u + b < \sigma$$

and

$$Q_{a,b}^{(\bar{x},\bar{y})} \neq 0 \text{ for some } a, b \in \mathbb{N} \text{ with } a/u + b = \sigma$$

*then the pair $(\bar{x}, \bar{y})$ is said to be a u-**zero of multiplicity** σ of Q. If $u = 1$ then $(\bar{x}, \bar{y})$ is a **zero of multiplicity** σ of Q.* □

Notice that for any pair $(\bar{x}, \bar{y}) \in \mathbb{F}_q^2$ and integers, $\sigma, u \in \mathbb{N}$ the requirement that a polynomial $Q \in \mathbb{F}_q[x, y] \backslash \{0\}$ has the pair as a u-zero of multiplicity at least σ can be written as $u\binom{\sigma+1}{2}$ homogeneous linear equations in the coefficients of Q.

8.6.2 Decoding algorithm

Suppose that a Generalized Reed-Solomon m-code, $RS(P, M, K)$, is used for communication where $P = \{P_1, \ldots, P_N\}$.

Suppose that $r = (r_1, \ldots, r_N) \in \mathcal{S}_M$ is some received word consisting of N chunks where the j'th chunk is $r_j = (r_{j,1}, \ldots, r_{j,M_j}) \in \mathcal{S}_{q,M_j}$ for $j = 1, \ldots, N$. Then let $u_j \in \{0, \ldots, M_j\}$ be maximal such that $r_{j,\alpha} \neq ?$ for $\alpha \leq u_j$. This means that u_j is the number of unerased symbols in the received chunk r_j.

Algorithm 8.27
Input: Code in use, $RS(P, M, K)$. Received word, $r \in \mathcal{S}_M$. Parameters, $\sigma \in \mathbb{N}\backslash\{0\}$ and λ satisfying $2 \leq \lambda \leq \lambda_{\max}$ with $\lambda_{\max}$ satisfying Inequality 8.24 for

$$i := 1 + \binom{\sigma + 1}{2} U \tag{8.25}$$

where $U := \sum_{j=1}^{N} u_j$ is the number of unerased symbols in the received word.
Output: Let

$$\ell := \left\lfloor \frac{i-1}{\lambda} + \frac{(\lambda - 1)(K-1)}{2} \right\rfloor = \left\lfloor \frac{\binom{\sigma+1}{2} U}{\lambda} + \frac{(\lambda - 1)(K-1)}{2} \right\rfloor \tag{8.26}$$

and let

$$T := \frac{\sum_{j=1}^{N} M_j + U}{2} - \frac{\ell}{\sigma}. \tag{8.27}$$

Then the output is the set of polynomials corresponding to codewords within distance T from the received word. That is the set

$$\{f \in \mathbb{F}_q[x] \mid \deg(f) < K \wedge \mathrm{d}_{\mathcal{S},M}(f(P, M), r) < T\}.$$

Notation: For $j = 1, \ldots, N$ let $r_j(x) := \sum_{\alpha=1}^{u_j} r_{j,\alpha}(x - P_j)^{\alpha-1}$.
Procedure:

- *Find $Q(x, y)$ such that $Q(x, y) \in \mathbb{F}_q[x, y]\backslash\{0\}$ and*

  1. *The pair $(x_j, \bar{y}_j(x))$ is a r_j-zero of multiplicity σ of Q for $j = 1, \ldots, N$.*

2. *The degree of y in Q is at most $\lambda - 1$ and $\deg^{(1,K-1)}(Q) \leq \ell$ where ℓ is given by Eqn. 8.26.*

- *Return the set*

$$\{f \in \mathbb{F}_q[x] \mid \deg(f) < K \wedge (y - f(x)) | Q \wedge \mathrm{d}_{\mathcal{S},M}(f(P,M), r) < T\}.$$

□

Proof:
To prove the correctness of the algorithm we must prove that the polynomial Q always exists and that it has the returned set of polynomials found from the y-linear factors of Q indeed contains all polynomials corresponding to codewords within distance T from the received word.

The requirement that each pair $(P_j, r_j(x))$ should be a u_j-zero of multiplicity σ of Q can be written as $i - 1$ (i given by Eqn. 8.25) homogeneous linear equations in the unknown coefficients of Q. By Lemma 8.25, the number of unknown coefficients of Q is at least i (unknown coefficients are those of terms where the degree of y is at most $\lambda - 1$ and the $(1, K-1)$ weighted degree is at most ℓ given by Eqn. 8.26). Therefore, by standard linear algebra, a non-zero solution exists.

Suppose that $f \in \mathbb{F}_q[x]$ where $\deg(f) < K$. Consider the polynomial $Q(x, f(x))$. Since $\deg(f) < K$ we have

$$\deg(Q(x, f(x))) \leq \deg^{(1,K-1)}(Q(x,y)) \leq \ell.$$

Furthermore, let $j \in \{1, \ldots, N\}$ and let $d_j := \mathrm{d}_{u_j}(f(P_j, u_j), r_j)$. Then $(x - P_j)^{u_j - d_j}$ divides $f(x) - r_j(x)$. Let

$$Q = \sum_{a,b} Q_{a,b}^{(P_j, r_j)} (x - P_j)^a (y - r_j(x))^b, \quad Q_{a,b}^{(P_j,r_j)} \in \mathbb{F}_q$$

then

$$Q(x, f(x)) = \sum_{a,b} Q_{a,b}^{(P_j, r_j)} (x - P_j)^a (f(x) - r_j(x))^b.$$

Since $Q_{a,b}^{(P_j,r_j)} = 0$ whenever $a/u_j + b < \sigma$ we have that $(x - P_j)^{\sigma(u_j - d_j)}$ divides $Q(x, f(x))$. All in all, a polynomial of degree $\sigma \sum_{j=1}^{N} (u_j - d_j)$ divides $Q(x, f(x))$. If this degree is larger than the degree of $Q(x, f(x))$

we can conclude that $Q(x, f(x)) = 0$ and, therefore, $y - f(x)$ is a factor of $Q(x, y)$. Notice that

$$\sigma \sum_{j=1}^{N}(u_j - d_j) = \sigma(U - \mathrm{d}_{\mathcal{S},M}(f(P, M), r) + \frac{\sum_{j=1}^{N} M_j - U}{2})$$
$$= \sigma(\frac{\sum_{j=1}^{N} M_j + U}{2} - \mathrm{d}_{\mathcal{S},M}(f(P, M), r)).$$

So

$$\sigma \sum_{j=1}^{N}(u_j - d_j) > \ell \Leftrightarrow \mathrm{d}_{\mathcal{S},M}(f(P, M), r)) < T$$

where T is given by Eqn. 8.27. Therefore, the output set contains all polynomials corresponding to codewords within $\mathrm{d}_{\mathcal{S},M}$-distance T from the received word. ■

Notice that the degree of Q in the algorithm is upper bounded by $\lambda - 1$. Therefore, Q can have at most $\lambda - 1$ factors in the form $y - f(x)$, so $\lambda - 1$ is also an upper bound on the number of polynomials in the output set.

This means that if we choose $\lambda = 2$ then the algorithm does unique decoding. In that case we should choose $\sigma = 1$ and thus we have

$$T = \frac{\sum_{j=1}^{N} M_j - U}{2} - \frac{U}{2} - \frac{K - 1}{2} = \frac{\sum_{j=1}^{N} M_j - K + 1}{2}.$$

If erasures cannot occur we have that $U = \sum_{j=1}^{N} M_j$ and thus

$$T = \sum_{j=1}^{N} M_j - \frac{\binom{\sigma+1}{2} \sum_{j=1}^{N} M_j}{\lambda\sigma} - \frac{(\lambda - 1)(K - 1)}{2\sigma}.$$

The algorithm can be efficiently implemented using Algorithm 5.11 to find Q satisfying the given conditions and then using the algorithm of Roth and Ruckenstein [33] or that of Gao and Shokrollahi [11] for finding factors of Q corresponding to codewords. For the case of Reed-Solomon codes an efficient implementation is described in Chapter 3.

For the cases of this correspondence an implementation has a worst case computational complexity of $O(\sigma^5 N^2)$.

Chapter 9

Unequal error protection and the m-metric

R. Refslund Nielsen[1]

Abstract

A new construction of codes composed from codes designed for the m-metric and codes with unequal error protection is presented and a lower bound on the minimum Hamming distance is found. A decoding method is given in terms of decoding methods for the component codes and it is shown that with suitable (and reasonable) assumptions on the error-correcting capability of the component decoders, the composed decoder corrects all error and erasure patterns of weight up to half the minimum distance bound where the weight of an erasure is 1/2.

9.1 Introduction

The m-metric and codes designed for this metric was first presented in [32]. Codes designed for this metric are useful in the situation where it can be said on beforehand that some positions of a received word are more likely to be erroneous than others.

Codes with unequal error protection are used in the case where some information symbols should be better protected than others. This means that even though too many errors occur for all the information symbols to be restored by the receiver, it should be possible to restore the information symbols of special importance.

The composed codes described in this text combines a code designed for the m-metric (referred to as an "m-metric code" below) with a code

[1]Department of Mathematics, Technical University of Denmark, Bldg 303 DK-2800 Lyngby Denmark

with unequal error protection in a construction similar to that of concatenation. The "outer" code is in this case the m-metric code and the code with unequal error protection has parameters such that the positions which are more critical to the m-metric code are better protected than the less critical positions. The effect is that with the composed code all positions are in principle equally well protected and an error in one position is no worse than an error in another position. The composed code is, therefore, designed for the Hamming distance and a lower bound on the minimum Hamming distance is given in terms of the distance properties of the component codes.

Furthermore, a decoding method is described which uses an error and erasure decoder for the m-metric code and either an error only or an error and erasure decoder for the code with unequal error protection depending on whether the overall decoder should be an error only or an error and erasure decoder. The extension of the m-distance of Definition 8.11 is used to describe precisely how erasures should be interpreted in the m-metric. It is shown that the composed decoder corrects all error patterns with weight up to half the lower bound on the minimum distance of the composed code.

An example of decoding a received word with the composed code is given using Algorithm 8.27.

9.2 The construction

This section describes a construction of composed codes, however, first the desired properties of the component codes are specified. In this paper $\mathbb{F}_q$ denotes a finite field with q elements and $\mathbb{N} = \{0, 1, 2, \dots\}$ denotes the set of non-negative integers. For any $n \in \mathbb{N}$ we let $\mathbb{F}_q^n$ denote the n-dimensional vector space over $\mathbb{F}_q$. If $v \in \mathbb{F}_q^n$ then subscripts in $\{1, \dots, n\}$ are used to denote each coordinate of v such that

$$v = (v_1, \dots, v_n)$$

with $v_i \in \mathbb{F}_q$ for $i = 1, \dots, n$.

9.2.1 Codes with unequal error protection

This subsection describes the properties of codes with unequal error protection that are needed in the composed code construction of this paper.

See [5] for a thorough description of codes with unequal error protection.

Consider an error-correcting code, $C_1 \subseteq \mathbb{F}_q^n$, with q^k codewords for integers n and k. Let $enc : \mathbb{F}_q^k \to C_1$ be a bijective mapping which encodes information (messages) into codewords and let $\mathrm{d} : \mathbb{F}_q^n \to \mathbb{N}$ denote the Hamming distance on $\mathbb{F}_q^n$.

The minimum Hamming distance, d, of C_1 is the minimum Hamming distance between two different codewords. That is

$$d = \min\{\mathrm{d}(w, y) \mid w, y \in C_1 \wedge w \neq y\}.$$

Alternatively, d can be seen as the smallest distance between the encoding of two different messages. That is

$$d = \min\{\mathrm{d}(\mathrm{enc}(u), \mathrm{enc}(v)) \mid u, v \in \mathbb{F}_q^k \wedge u \neq v\}.$$

However, let $u \in F_q^k$ and let $u_{1;i} := (u_1, \ldots, u_i)$ denote the projection of u on the first i positions for $i = 1, \ldots, k$. Then let

$$d_{1;i} = \min\{\mathrm{d}(\mathrm{enc}(u), \mathrm{enc}(v)) \mid u, v \in \mathbb{F}_q^k \wedge u_{1;i} \neq v_{1;i}\}.$$

So $d_{1;i}$ is the minimum distance between the encoding of two messages whose projection on the i first positions differ. In all cases, $d_{1;i} \geq d$ and $d_{1;k} = d$. If $d_{1;i} > d$ then the i first symbols of the message are better protected than the message as a whole because more errors must occur for that part of the message to be unrecoverable by the receiver. In this case C_1 is called a code with **unequal error protection**.

Suppose that C_1 has the following property for some fixed positive rational number, λ, and all $i = 1, \ldots, k$:

$$d_{1;i} \geq \lambda(k + 1 - i).$$

Then we will say that C is a code with unequal error protection with parameters $(n, k, \lambda(k \ldots 1))$. If $\lambda = 1$ we may omit λ in the description of the parameters and just write $(n, k, k \ldots 1)$.

Example 9.1

The following are generator and parity check matrices for a binary code

with unequal error protection and parameters $(9, 3, 2(3 \ldots 1))$*:*

$$H_1 = \begin{bmatrix} 0 & 1 & 1 & \\ 1 & 1 & 0 & \\ 1 & 1 & 0 & \\ 1 & 0 & 0 & I_{6\times 6} \\ 1 & 0 & 0 & \\ 1 & 0 & 0 & \end{bmatrix} \qquad G_1 = \begin{bmatrix} & 0 & 1 & 1 & 1 & 1 & 1 \\ I_{3\times 3} & 1 & 1 & 1 & 0 & 0 & 0 \\ & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

That this code has the specified parameters is seen from the parity check matrix. Any sum of 5 *or fewer columns including the first column is non-zero. Any sum of* 3 *or fewer columns including the second column is non-zero. Finally, the third column is non-zero.*

Let $q > 3$ *be a prime power and let* α *be a primitive element of* $\mathbb{F}_q$*. Then the following are generator and parity check matrices for a* $(8, 3, 2(3 \ldots 1))$ *code with unequal error protection over* $\mathbb{F}_q$*:*

$$H_1' = \begin{bmatrix} 1 & 1 & 1 & \\ \alpha & 1 & 0 & \\ \alpha^2 & 1 & 0 & I_{6\times 6} \\ 1 & 0 & 0 & \\ 1 & 0 & 0 & \end{bmatrix} \qquad G_1' = \begin{bmatrix} & 1 & \alpha & \alpha^2 & 1 & 1 \\ I_{3\times 3} & 1 & 1 & 1 & 0 & 0 \\ & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

□

Consider the parity check matrix, H, of a linear $(n, k, \lambda(k \ldots 1))$ code with unequal error protection and reduce it to the form $H' = [A|I]$ where I is a $(n-k) \times (n-k)$ identity matrix. Assume without loss of generality that the reduction is done without exchanging columns. Then the first column of A must have Hamming weight at least $\lambda k - 1$ so we have the following analogue to the usual MDS bound

$$n - k \geq \lambda k - 1 \Leftrightarrow n \geq (\lambda + 1)k - 1. \tag{9.1}$$

However, notice that this is a bound for linear codes only and there is no immediate analogue to the usual non-linear MDS bound. So it is an open question if non-linear codes with unequal error-protection exists which are better than codes meeting the bound of Exp. 9.1.

9.2.2 The m-metric

Let m be a given positive integer and define the m-distance (see Theorem 6.5) between two vectors, $a, b \in \mathbb{F}_q^m$ as follows:

$$\mathrm{d}_m(a,b) = \begin{cases} 0 & \text{if } a = b \\ m+1-j & \text{otherwise, where } j = \min\{j \mid a_j \neq b_j\}. \end{cases} \tag{9.2}$$

This is extended to a distance on $\mathbb{F}_q^{mN}$ for any positive integer, N by seeing vectors in $\mathbb{F}_q^{mN}$ as consisting of N chunks with m elements in each and then take the total m-distance to be the sum of the distance between each chunk. That is

$$\mathrm{d}_m(u,v) = \sum_{j=1}^{N} \mathrm{d}_m(u_j, v_j)$$

where $u, v \in (\mathbb{F}_q^m)^N$ and $u_j, v_j \in \mathbb{F}_q^m$ for $j = 1, \ldots, N$.

An alternative way to see the m-metric is to consider the elements of vectors $a, b \in \mathbb{F}_q^m$ as coefficients of polynomials,

$$a(x) = \sum_{j=1}^{m} a_j x^{j-1} \text{ and } b(x) = \sum_{j=1}^{m} b_j x^{j-1}.$$

If $a \neq b$ then the m-distance between a and b is $m+1-j$ where j is the largest integer such that x^{j-1} divides $a(x) - b(x)$.

9.2.3 The composed codes

Let $C_2 \subseteq \mathbb{F}_q^{mN}$ be a code of length mN containing q^K codewords with minimum m-distance D and let $C_1 \subseteq \mathbb{F}_q^n$ be a code with unequal error protection with parameters $(n, m, \lambda(m \ldots 1))$. Then construct the following composed code:

$$C := \{(\mathrm{enc}(u_1), \ldots, \mathrm{enc}(u_N)) \mid (u_1, \ldots, u_N) \in C_2\} \tag{9.3}$$

where $\mathrm{enc} : \mathbb{F}_q^m \to C_1$ encodes m symbols into a codeword of C_1. In the following, C_1 will be called the **inner code** of C and C_2 will be called the **outer code** of C.

It is clear that C has q^K codewords of length nN. Furthermore, we have the following theorem:

Theorem 9.2
The minimum Hamming distance of C (defined in Eqn. 9.3) is at least λD. □

Proof:
Let $U = (\mathrm{enc}(u_1), \ldots, \mathrm{enc}(u_N))$ and $V = (\mathrm{enc}(v_1), \ldots, \mathrm{enc}(v_N))$ be two different codewords of C. Then $u = (u_1, \ldots, u_N)$ and $v = (v_1, \ldots, v_N)$ are different codewords of C_2 and thus have m-distance at least D.

Let $j \in \{1, \ldots, N\}$ be arbitrary. There are now two cases:

- Suppose that $u_j \neq v_j$ and let i denote the m-distance between u_j and v_j. That is $\mathrm{d}_m(u_j, v_j) = i > 0$. By Eqn. 9.2 this means that $u_{j,1;m+1-i} \neq v_{j,1;m+1-i}$ and by the parameters of C_1 we then have

$$\mathrm{d}(\mathrm{enc}(u_j), \mathrm{enc}(v_j)) \geq \lambda(m + 1 - (m + 1 - i)) = \lambda i = \lambda \mathrm{d}_m(u_j, v_j).$$

- Suppose that $u_j = v_j$. Then $\mathrm{d}(\mathrm{enc}(u_j), \mathrm{enc}(v_j)) = 0 = \lambda \mathrm{d}_m(u_j, v_j)$.

So all in all we have

$$\mathrm{d}(U, V) = \sum_{j=1}^{N} \mathrm{d}(\mathrm{enc}(u_j), \mathrm{enc}(v_j)) \geq \sum_{j=1}^{N} \lambda \mathrm{d}_m(u_j, v_j) = \lambda \mathrm{d}_m(u, v) \geq \lambda D.$$

■

Example 9.3
The following is the generator matrix of a binary $(9, 4)$ code with minimum 3-distance 6 (a Reed-Solomon m-code, see Definition 6.1):

$$G_2 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Using this as the outer code and the code generated by G_1 in Ex. 9.1 as the inner code in Eqn. 9.3 gives a binary $(27, 4, \geq 12)$ code by Theorem 9.2. By examining the code it is found that the minimum Hamming distance actually equals 12. □

Notice that the code in the example above is not particularly good. For example, we can get a binary $(21, 4, 12)$ code as a product code of a binary

$(7, 3, 4)$ simplex code and a binary $(3, 1, 3)$ repetition code. It turns out that it is quite difficult to come up with a code using the construction in Eqn. 9.3 which is not directly inferior to some product code, however, it is an open question if this is an intrinsic property of the construction or because of lack of good component codes ("good" here means with parameters close to some upper bound). — The bound in Exp. 9.1 seems to prevent good composed codes to be constructed based on linear codes with unequal error protection.

9.3 Decoding

This section presents a decoding method for error and erasure decoding up to half the designed minimum distance of the code in Eqn. 9.3 for certain values of λ. The decoding method assumes that decoding methods are already available for the inner and outer codes.

The method operates entirely in $\mathbb{F}_q$ and since the code construction is similar to that of concatenation (where several decoding attempts are needed unless list decoding techniques are used, see Section 7.4 and Section 7.6) it is a bit surprising that only one attempt is necessary.

An example is given in the next section using Algorithm 8.27.

9.3.1 Extended distances

In order to describe decoding with erasures it is convenient to extend Hamming distance and m-distance to a domain containing a symbol, ?, which denotes an erasure. This is done in Section 8.3. The definitions are summarized below. That the distances are actually distances follows from Lemma 8.6 and Theorem 8.12.

Let $\mathcal{A}_q := \mathbb{F}_q \cup \{?\}$ and define the function $\delta_{\mathcal{A}} : \mathcal{A}_q \times \mathcal{A}_q \to \{0, 1/2, 1\}$ by

$$\delta_{\mathcal{A}}(a, b) := \begin{cases} 0 & \text{if } a = b \\ 1/2 & \text{if } a \neq b \text{ and } ? \in \{a, b\} \\ 1 & \text{otherwise.} \end{cases} \tag{9.4}$$

We then have the following definition (Definition 8.7):

Definition 9.4 (Extended Hamming distance and weight)
*The **extended Hamming distance** on $\mathcal{A}_q^n$ is the function $\mathrm{d}_{\mathcal{A}} : (\mathcal{A}_q^n)^2 \to \mathbb{N}/2$ (where $\mathbb{N}/2 = \{0, 1/2, 1, 3/2, \dots\}$) given by*

$$\mathrm{d}_{\mathcal{A}}(u, v) = \sum_{i=1}^{n} \delta_{\mathcal{A}}(u_i, v_i) \tag{9.5}$$

where δ is defined in Eqn. 9.4.

*The **extended Hamming weight** of a tuple, $u \in \mathcal{A}_q^n$, is given by $\mathbf{w}_{\mathcal{A}}(u) := \mathrm{d}_{\mathcal{A}}(u, 0)$.* □

In order to make a similar extension of the m-distance with properties convenient for this paper, it turns out that we need to consider only the set

$$\mathcal{S}_q^m := \{(a_1, \dots, a_m) \in \mathcal{A}_q^m \mid a_\alpha =? \Rightarrow a_{\alpha+1} =? \text{ for } \alpha = 1, \dots, m-1\}. \tag{9.6}$$

Furthermore, let $s : \mathcal{S}_q^m \to \{1, \dots, m+1\}$ be given by

$$s(a) := \begin{cases} m+1 & \text{if } a \in \mathbb{F}_q^m \\ \alpha & \text{if } a \notin \mathbb{F}_q^m \text{ and } \alpha = \min\{\alpha \mid a_\alpha =?\} \end{cases}$$

for any $a = (a_1, \dots, a_m) \in \mathcal{S}_q^m$.

We then have the following definition (Definition 8.11):

Definition 9.5
*The **extended m-distance** between two tuples, $a, b \in \mathcal{S}_q^m$ (Eqn. 9.6), is given as follows where $\alpha := \min\{s(a), s(b)\}$, $\beta := \max\{s(a), s(b)\}$, $j := \min\{j \mid a_j \neq b_j\}$, $a = (a_1, \dots, a_m)$, and $b = (b_1, \dots, b_m)$:*

$$\mathrm{d}_{\S,m}(a, b) := \begin{cases} 0 & \text{if } a = b \\ \alpha - j + \frac{\beta - \alpha}{2} = \frac{s(a)+s(b)}{2} - j & \text{if } a \neq b. \end{cases} \tag{9.7}$$

□

This is extended to a distance on $(\mathcal{S}_q^m)^N$ for any positive integer, N by seeing tuples in $(\mathcal{S}_q^m)^N$ as consisting of N chunks with m elements in each and then taking the total extended m-distance to be the sum of the extended distances between each chunk. That is

$$\mathrm{d}_{\mathcal{S},m}(u, v) = \sum_{j=1}^{N} \mathrm{d}_{\mathcal{S},m}(u_j, v_j)$$

where $u, v \in (\mathcal{S}_q^m)^N$ with $u_j, v_j \in \mathcal{S}_q^m$ for $j = 1, \ldots, N$.

9.3.2 Inner and outer decoder

Let C be a composed code constructed as in Eqn. 9.3. Suppose that a codeword, $\hat{c} \in C \subseteq (\mathbb{F}_q^n)^N$, is sent and $v \in (\mathcal{A}_q^n)^N$ is received.

The first step in the decoding procedure is to decode each of the inner words, v_j. For this we will assume that some decoder (referred to as "the inner decoder" in the rest of the paper) is available for the inner code which — in principle — does the following:

Remember that C_1 is a $(n, m, \lambda(m \ldots 1))$ code with unequal error protection. If there is no inner codeword, $c_j \in C_1$, with distance $\mathrm{d}_{\mathcal{A}}(c_j, v_j) < \lambda m/2$ then let $w_j := \lambda m/2$. Otherwise, let c_j be an arbitrary codeword with minimal distance to v_j, let $w_j := \mathrm{d}_{\mathcal{A}}(c_j, v_j)$, and let $y_j = (y_{j,1}, \ldots, y_{j,m}) := \mathrm{enc}^{-1}(c_j)$. In the following we will refer to w_j as the **observed error weight** in the j'th inner word.

Finally, mark erasures by letting $y_{j,m-\lfloor 2w_j/\lambda \rfloor + 1}, \ldots, y_m :=?$ and output y_j. The idea of this is to keep only those information symbols which are agreed on by every codeword with minimal distance to the received word.

Notice that if the received word, v, is not allowed to contain erasures then the inner decoder need not be able to handle erasures in the input, however, erasures should still be marked appropriately in the output.

The outer decoder must in all cases be an error and erasure decoder which given a received word, $y \in \mathcal{S}_q^{mN}$, finds the unique codeword, $c \in C_2 \subseteq \mathbb{F}_q^{mN}$ (if it exists), such that

$$\mathrm{d}_{\mathcal{S},m}(y, c) < \frac{D}{2} \tag{9.8}$$

where D is the minimum m-distance of C_2. If no such codeword exists, the decoder should report a failure. — This makes sense since the extended m-distance is an extension of the m-distance to $\mathcal{S}_q^{mN}$. Therefore, the minimum extended m-distance of C_2 equals the minimum m-distance.

9.3.3 Composed decoder

The composed decoding procedure is then as follows:

Algorithm 9.6
Given the received word, v, decode each inner word, v_j, for $j = 1, \ldots, N$ giving the result y_j using the inner decoder. Form the outer word, $y = (y_1, \ldots, y_N)$, and decode it using the outer decoder which results in either a failure or a codeword, c, which is given as output. □

Let the true error weight in the j'th inner word be given by

$$\hat{w}_j := \mathrm{d}_{\mathcal{A}}(v_j, \hat{c}_j)$$

where $\hat{c} = (\hat{c}_1, \ldots, \hat{c}_N) \in C$ is the sent codeword. The total error weight is then

$$\hat{W} := \sum_{j=1}^{N} \hat{w}_j.$$

The following theorem gives the error-correcting capability of Algorithm 9.6. Notice that the second condition on the error pattern is always satisfied if $\lambda = 1$ or if $\lambda = 2$ and erasures are not allowed in the received word.

Theorem 9.7
Algorithm 9.6 recovers the correct codeword, $\hat{c}$, if

$$\hat{W} < \frac{\lambda D}{2}$$

and if $2w_j/\lambda \in \mathbb{N}$ for $j = 1, \ldots, N$ where w_j is the observed error weight in the j'th inner word. □

Proof:
Suppose that $2w_j/\lambda \in \mathbb{N}$ for all $j = 1, \ldots, N$. Let $j \in \{1, \ldots, N\}$ and let $s_j := 2w_j/\lambda$ denote the number of erasures in y_j. There are two cases:

- Suppose that $y_{j,\alpha} = \hat{c}_{j,\alpha}$ for all $\alpha < m + 1 - s_j$ — this means that all the unerased information symbols in the output from the j'th inner decoding are correct. We then have

  $$\mathrm{d}_{\mathcal{S},m}(y_j, \hat{c}_j) = \frac{m + 1 + (m + 1 - s_j)}{2} - (m + 1 - s_j) = \frac{s_j}{2} = \frac{w_j}{\lambda} \leq \frac{\hat{w}_j}{\lambda}$$

  where the last inequality follows from the fact that $\hat{w}_j \geq w_j$ in all cases since w_j is at most the minimal extended Hamming distance

between the j'th received inner word and any inner codeword while $\hat{w}_j$ is the distance between the j'th received word and the sent inner codeword.

- Suppose that $y_{j,\alpha'} \neq \hat{c}_{j,\alpha'}$ for some $\alpha' < m + 1 - s_j$ and let α be the smallest value such that $y_{j,\alpha} \neq \hat{c}_{j,\alpha}$. This means that the j'th inner decoding changed the received word into a wrong codeword, c, and since the α'th information element is protected by at least distance $\lambda(m + 1 - \alpha)$ this means that

$$\lambda(m + 1 - \alpha) \leq \mathrm{d}(c, \hat{c}) \leq \hat{w}_j + w_j \Rightarrow \alpha \geq m + 1 - \frac{\hat{w}_j + w_j}{\lambda}.$$

  This gives

$$\mathrm{d}_{\mathcal{S},m}(y_j, \hat{c}_j) = \frac{m + 1 + (m + 1 - s_j)}{2} - \alpha \leq \frac{\hat{w}_j + w_j}{\lambda} - \frac{s_j}{2} = \frac{\hat{w}_j}{\lambda}.$$

In both cases we have

$$\lambda \mathrm{d}_{\mathcal{S},m}(y_j, \hat{c}_j) \leq \hat{w}_j.$$

So

$$\lambda \mathrm{d}_{\mathcal{S},m}(y, \hat{c}) = \sum_{j=1}^{N} \lambda \mathrm{d}_{\mathcal{S},m}(y_j, \hat{c}_j) \leq \sum_{j=1}^{N} \hat{w}_j = \hat{W}.$$

The theorem then follows by the assumption on the error-correcting capability of the outer decoder in Eqn. 9.8. ■

9.4 Example

Let C_1 be the linear $(7, 4, 4 \ldots 1)$ code with unequal error protection over $\mathbb{F}_3$ which has the following generator matrix:

$$\left[\begin{array}{cccc} & 1 & 2 & 1 \\ & 1 & 1 & 0 \\ I_{4\times 4} & 1 & 0 & 0 \\ & 0 & 0 & 0 \end{array}\right]$$

and let C_2 be the linear $(12, 4)$ code over $\mathbb{F}_3$ with the following generator matrix:

$$\left[I_{4\times 4} \begin{array}{cccccccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 2 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 2 & 0 & 0 & 1 \end{array} \right]$$

Notice that C_2 is a Reed-Solomon m-code (Definition 6.1) with minimum 4-distance 9.

Let C be given as in Eqn. 9.3. Then C is a $(21, 4, \geq 9)$ code over $\mathbb{F}_3$.

The information $i = (1, 0, 1, 2)$ encodes into the following codeword of C_2:

$$1012\ 1212\ 0112$$

which is encoded into the following codeword of C:

$$\hat{c} := 1012221\ 1212111\ 0112210.$$

Suppose that $\hat{c}$ is sent, but the following is received:

$$v := 10022?1\ 1012211\ 011?210.$$

This represents an error pattern with 3 errors and 2 erasures,

The minimal distance from each of these 3 inner words to a codeword in C_1 is 3/2, 1, and 1/2 respectively. The inner decoder gives the following (the words in curly braces are the inner codewords with minimal distance to the received inner word):

$$\begin{aligned} 10022?1 &\longrightarrow \left\{ \begin{array}{c} 1002121 \\ 1012221 \\ 1102201 \end{array} \right\} \longrightarrow 1??? \\ 1012211 &\longrightarrow \{\ 1012221\ \} \longrightarrow 10?? \\ 011?210 &\longrightarrow \left\{ \begin{array}{c} 0110210 \\ 0111210 \\ 0112210 \end{array} \right\} \longrightarrow 011? \end{aligned}$$

As the outer decoder we use Algorithm 8.27 with $\lambda = 2$ and $\sigma = 1$. Using the notation of the algorithm we have $U = 1 + 2 + 3 = 6$, so we have to

find $Q = Q_0(x) + y \cdot Q_1(x) \neq 0$ such that $\deg(Q_0) \leq 4$ and $\deg(Q_1) \leq 1$ and where $(0, 1)$ is a 1-zero, $(1, 1)$ is a 2-zero, and $(2, (x-2) + (x-2)^2)$ is a 3-zero (Definition 8.26).

The possibly non-zero coefficients of Q are $Q_{0,0}$, $Q_{1,0}$, $Q_{2,0}$, $Q_{3,0}$, $Q_{4,0}$, $Q_{0,1}$, and $Q_{1,1}$. Using Eqn. 5.11 we get that the coefficients must satisfy

$$\begin{array}{rcccccccccc}
Q_{0,0}^{(0,1)} = & & Q_{0,0} & & & & & +Q_{0,1} & & = 0 \\
Q_{0,0}^{(1,1)} = & & Q_{0,0} & +Q_{1,0} & +Q_{2,0} & +Q_{3,0} & +Q_{4,0} & +Q_{0,1} & & = 0 \\
Q_{1,0}^{(1,1)} = & & & Q_{1,0} & +2Q_{2,0} & & +Q_{4,0} & & +Q_{1,1} & = 0 \\
Q_{0,0}^{(2,(x-2)+(x-2)^2)} = & & Q_{0,0} & +2Q_{1,0} & +Q_{2,0} & +2Q_{3,0} & +Q_{4,0} & & & = 0 \\
Q_{1,0}^{(2,(x-2)+(x-2)^2)} = & & & Q_{1,0} & +2Q_{2,0} & & +Q_{4,0} & & +2Q_{1,1} & = 0 \\
Q_{2,0}^{(2,(x-2)+(x-2)^2)} = & & & & Q_{2,0} & & & +Q_{0,1} & +2Q_{1,1} & = 0
\end{array}$$

One solution to this system of equations is

$$Q = 2 + x^2 + x^3 + y = y - (1 + 0x + 1x^2 + 2x^3).$$

Notice that the coefficients to the right, 1012, equals the information that was originally encoded into the sent codeword. The decoding method, therefore, successfully recovered the sent information.

Bibliography

[1] D. Augot and L. Pecquet: "A Hensel lifting to replace factorization in list-decoding of algebraic-geometric and Reed-Solomon codes" *IEEE Transactions on Information Theory*, vol. 46, no. 7, pp. 2605-2614, 2000.

[2] R. E. Blahut: "Theory and Practice of Error Control Codes" Addison-Wesley Publishing Company, 1983.

[3] E. L. Blokh and V. V. Zyablov, "Generalized concatenated codes", (in Russian), *Svyaz'*, Moscow, 1976.

[4] A. Brouwer: Bounds on the minimum distance of linear codes. Online. URL: http://www.win.tue.nl/~aeb/voorlincod.html.

[5] E. Englund: "Codes with Unequal Error Protection" *Ph.D. Dissertation No. 412*, Department of Electrical Engineering, Linköping University, 1995.

[6] T. Ericson, "A simple analysis of the Blokh-Zyablov decoding algorithm" *Lecture Notes in Computer Science 307*, pp. 43-57, Springer-Verlag, 1988.

[7] G.-L. Feng and K.K. Tzeng: "A Generalization of the Berlekamp-Massey Algorithm for Multisequence Shift-Register Synthesis with Applications to Decoding Cyclic Codes", *IEEE Transactions on Information Theory*, vol. 37, pp. 1274-1287, Sep. 1991.

[8] W. Feng and R. E. Blahut: "Some Results on the Sudan Algorithm", *Proceedings 1998 IEEE International Symposium on Information Theory, p.57*, august 1998.

[9] G. D. Forney, Jr., "Concatenated Codes" *M.I.T. Press*, Cambridge, MA, 1966.

[10] G. D. Forney, Jr., "Generalized Minimum Distance Decoding" *IEEE Transactions on Information Theory*, vol. 12, no. 2, pp. 125-131, 1966.

[11] S. Gao and M.A. Shokrollahi: "Computing roots of polynomials over function fields of curves" *D. Joyner, ed.: Coding Theory and Cryptography*, pp. 114-128. Springer-Verlag, 1999.

[12] K. Geddes, S. Czapor, and G. Labahn: "Algorithms for Computer Algebra" *Kluwer Academic Publishers*, 1992.

[13] V. Guruswami and M. Sudan, "Improved Decoding of Reed-Solomon and Algebraic-Geometric Codes" *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 1757-1767, 1999.

[14] V. Guruswami and M. Sudan, "List Decoding Algorithms for certain Concatenated Codes", preprint, 2000.

[15] T. Høholdt, J.H. van Lint, and R. Pellikaan: "Handbook of Coding Theory, Chapter 10" (edited by V.S. Pless and W.C. Huffman), Elsevier Science B.V., 1998.

[16] T. Høholdt and R. Refslund Nielsen: "Decoding Hermitian codes with Sudan's algorithm" *M. Fossorier, H. Imai, S. Lin, A. poli (eds.): Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, Lecture Notes in Computer Science 1719, pp. 260-270, Springer-Verlag, 1999.

[17] H. Elbrønd Jensen, R. Refslund Nielsen, and T. Høholdt: "Performance analysis of a decoding algorithm for algebraic geometry codes" *IEEE Trans. on Inform. Theory, vol. 45, pp. 1712-1717*, July 1999.

[18] J. Justesen, "A Class of Constructive Asymptotically Good Algebraic Codes" *IEEE Transactions on Information Theory*, vol. 18, no. 5, pp. 652-656, 1972.

[19] J. Justesen and T. Høholdt, "Bounds on List Decoding of MDS Codes", *IEEE Transactions on Information Theory*, vol. 47, no. 4, pp. 1604-1609, 2001.

[20] R. Kötter: "On Algebraic Decoding of Algebraic-Geometric and Cyclic Codes" *Ph.D. Dissertation No. 419*, Department of Electrical Engineering, Linköping University, 1996.

[21] R. Kötter and A. Vardy, "Algebraic Soft-Decision Decoding of Reed-Solomon Codes", Preprint, 2000.

[22] R. Lidl and H. Niederreiter: "Finite fields" *Addison-Wesley*, 1983.

[23] J.H. van Lint: "Introduction to Coding Theory", Springer-Verlag, 1982.

[24] F. J. MacWilliams and N. J. A. Sloane, "The Theory of Error-Correcting Codes", Elsevier Science B.V., 1977.

[25] M. Morii and M. Kasahara: "Generalized Key-Equation of Remainder Decoding Algorithm for Reed-Solomon Codes" *IEEE Trans. Inform. Theory*, vol 38, no. 6, pp. 1801-1807, Nov. 1992.

[26] R. Refslund Nielsen: "Decoding AG-codes beyond half the minimum distance", *Master's Thesis*, Technical University of Denmark, Aug. 1998.

[27] R. Refslund Nielsen: "A Class of Sudan-Decodable Codes", *IEEE Transactions on Information Theory*, pp. 1564-1572, vol. 46, No. 4, July 2000.

[28] R. Refslund Nielsen and T. Høholdt, "Decoding Reed-Solomon Codes Beyond Half the Minimum Distance" *Buchmann et al: Coding Theory, Cryptography and Related Areas*, Springer, pp. 221-236, 2000.

[29] S.M. Reddy, "Decoding Iterated Codes" *IEEE Transactions on Information Theory*, vol. 16, no. 5, pp. 624-627, 1970.

[30] S.M. Reddy and J.P. Robinson, "Random Error and Burst Correction by Iterated Codes" *IEEE Transactions on Information Theory*, vol. 18, no. 1, pp. 182-185, 1972.

[31] I.S. Reed and G. Solomon, "Polynomial codes over certain finite fields" *SIAM Journal of Applied Mathematics*, vol. 8, pp. 300-304, 1960.

[32] M. Yu. Rosenbloom and M. A. Tsfasman: "Codes for the m-metric" *Problems of Information Transmission*, vol. 33, no. 1, pp. 45-52, 1997.

[33] R. M. Roth and G. Ruckenstein, "Efficient Decoding of Reed-Solomon Codes Beyond Half the Minimum Distance", *IEEE Transactions on Information Theory*, vol. 46, no. 1, pp. 246-257, 2000.

[34] S. Sakata, H. Elbrønd Jensen and T. Høholdt: "Generalized Berlekamp-Massey Decoding of Algebraic-Geometric Codes up to half the Feng-Rao Bound" *IEEE Trans. Inform. Theory*, vol 41, no. 6, pp. 1762-1768, Nov. 1995.

[35] M.A. Shokrollahi and H. Wassermann: "List Decoding of Algebraic-Geometric Codes" *IEEE Trans. Inform. Theory, vol 45, pp. 432-437*, March 1999.

[36] W.R. Smythe, Jr. and L.A. Johnson, "Introduction to Linear Programming, with Applications" *Prentice-Hall Inc.*, 1966.

[37] H. Stichtenoth: "Algebraic function fields and codes" *Springer-Verlag*, 1993.

[38] M. Sudan: "Decoding of Reed Solomon Codes beyond the Error-correction bound" *Journal of complexity 13*, pp. 180-193, 1997.

[39] E.J. Weldon, Jr., "Decoding Binary Block Codes on Q-ary Output Channels" *IEEE Transactions on Information Theory*, vol. 17, no. 6, pp. 713-718, 1971.

[40] S. B. Wicker and V. K. Bhargava: "Reed-Solomon Codes and Their Applications" *IEEE Press*, section 5.3, p.87, 1994.

[41] X.-W. Wu and P.H. Siegel, "Fast Computation of Roots of Polynomials over Function Fields", preprint, 2000.

[42] C. Xing and S. Ling: "A Class of Linear Codes with Good Parameters", *IEEE Transactions on Information Theory*, vol. 46, no. 6, pp. 2184-2188, 2000.

[43] C. Xing, H. Niederreiter, and K.Y. Lam, "A Generalization of Algebraic-Geometry Codes", *IEEE Transactions on Information Theory*, vol. 45, no. 7, pp. 2498-2501, 1999.

List of Figures

Index